IBM Tealeaf CX UI Capture j2
Version 3 Release 1
June 12, 2014

IBM Tealeaf CX UI Capture j2 Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 77.

This edition applies to version 3, release 1, modification 0 of IBM Tealeaf CX UI Capture j2 and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1999, 2014.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

IBM Tealeaf CX UI Capture j2 Guide . . . v

Chapter 1. IBM Tealeaf CX UI Capture j2 1
Working with use cases . . . 1
Validating use cases with Client-Side Capture . . . 2
System prerequisites . . . 2
Supported browsers . . . 2
How CX UI Capture j2 for replay works . . . 3
How CX UI Capture j2 captures interactions . . . 4
How CX UI Capture j2 replays captured interactions . . . 5
Related documentation . . . 7

Chapter 2. CX UI Capture j2 installation and implementation. . . . 9
Install version . . . 9
Support for CX RealiTea Viewer . . . 9
Support for mobile web application capture . . . 9
CX Passive Capture Application configuration . . . 9
Installation and deployment plan . . . 10
Stage 1: Development environment . . . 10
Stage 2: Testing environment . . . 11
Stage 3: Production environment . . . 14
Configuring the JavaScript . . . 15
CX UI Capture j2 JavaScript interactions with web application pages . . . 15
Block sensitive data fields . . . 15
Installation on web pages. . . . 16
Unique HTML IDs . . . 16
Cookies. . . . 16
Installation on the web server . . . 16
References to the JavaScript file. . . . 16
IBM Tealeaf target page . . . 17
Configuration of CX UI Capture j2 JavaScript . . . 18
IIS configuration. . . . 19
Non-IIS web server configuration . . . 19
Web page modifications . . . 19
Change management for Document Object Model elements . . . 20
Upgrade CX UI Capture j2 . . . 20
Support for legacy headers . . . 21
Uninstalling CX UI Capture j2 . . . 21

Chapter 3. CX UI Capture j2 Configuration Wizard . . . 23
Supported browsers . . . 23
Configuration Wizard controls . . . 23
Build types . . . 23
Step 1: Browser service configuration . . . 24
Step 2: Queue service configuration . . . 25
Step 3: Message service configuration (privacy masking configuration) . . . 26
Step 4: Serializer. . . . 27
Step 5: Modules . . . 28
Step 6: Miscellaneous settings . . . 30
RegEx Tester . . . 30

Chapter 4. CX UI Capture j2 usage guidelines . . . 33
Application scope . . . 33
Supported application types . . . 33
Supported protocols . . . 33
Before you begin development . . . 34
IBM Tealeaf access . . . 34
Third-party content. . . . 34
Personnel . . . 34
Development considerations. . . . 35
Development cycle . . . 35
When to deploy CX UI Capture j2 during development . . . 35
Development and Test environments . . . 35
Performance management . . . 36
Application content. . . . 36
Unique identifiers . . . 36
Do not use Document Object Model keywords as form field names . . . 36
Application objects . . . 37
Private data . . . 38
Frequency of posts . . . 38
UI events overridden by CX UI Capture j2 . . . 38
CX UI Capture j2 deployment . . . 39
Placement of CX UI Capture j2 . . . 39
JavaScript . . . 39
Create custom events . . . 39
Next steps . . . 40

Chapter 5. CX UI Capture j2 reference 41
Core configuration . . . 41
Core. . . . 43
Configure client events by module . . . 44
Services configuration . . . 46
Queue service configuration . . . 48
Browser service configuration . . . 49
Message service configuration . . . 50
Serializer service configuration . . . 53
Modules configuration. . . . 54
Performance . . . 55
localStorage configuration . . . 55

Chapter 6. CX UI Capture j2 Public API Reference . . . 57
TLT.init(object configObject) . . . 57
TLT.rebind(/*optional*/ DOMElement root) . . . 57
TLT.flushAll(void) . . . 58
TLT.setAutoFlush(AutoFlushFlag flag) . . . 58
TLT.processDOMEvent(DOMEvent event) . . . 58
TLT.logCustomEvent(DOMString name, object customMsgObj) . . . 59

© Copyright IBM Corp. 1999, 2014

TLT.logExceptionEvent(DOMString msg, /*optional*/ DOMString url, /*optional*/ long line) 59
TLT.logScreenviewLoad(DOMString name, /*optional*/ DOMString referrerName, /*optional*/ DOMElement root) 59
TLT.logScreenviewUnload(DOMString name) . . . 60
TLT.getSessionData() 60
TLT.registerBridgeCallbacks 60
TLT.logScreenCapture 62

Chapter 7. CX UI Capture j2 FAQ . . . 63
Getting started 63
What is CX UI Capture j2? 63
What versions of UI Capture are available? . . . 63
Why do I need CX UI Capture j2? 63
How do I implement CX UI Capture j2? 64
What is the size of CX UI Capture j2? 65
Can I bundle the CX UI Capture j2 library with other JavaScripts? 65
Will CX UI Capture j2 impact the performance or behavior of my website? 65
Where can I get latest version of CX UI Capture j2? 66
What is being captured by CX UI Capture j2? . . 66
I am having trouble with CX UI Capture j2 implementation. Where do I get support or help?. 66
Using CX UI Capture j2 66
How do I configure what is captured by CX UI Capture j2? 66
What is captured from mobile web sessions? . . 66
What if I find a bug with CX UI Capture j2? . . 67
How can I create events from CX UI Capture j2 data? 67
How can I search for CX UI Capture j2 data? . . 67
How do I capture only `mouseover` events on menu? 67
How do I capture `keyup` events? 67
How do I report on client-side validation/error messages? 67
How do I force the sending of queued events? . 68
For the first phase, the only desired metric is the Page Render time. Are there configuration settings that limit POSTs to just that event?. . . 68
I use DHTML on my checkout form. Is CX UI Capture j2 going to capture the dynamically rendered DOM elements?. 68
How do I get render times from CX UI Capture j2? 69
Do I need to have IDs assigned to my DOM elements that I would like to capture? 69
Can I customize CX UI Capture j2 JavaScripts? 69
How come I get cross-domain error messages in my browsers debug console related to iFrames on my site? 69
Upgrading the library 70
How do I determine which version of the library I am using? 70
How do I determine which version of the `TealeafTarget.jsp` file I am using? 70
When do I upgrade? 70
How do I find out about available updates? . . 70

Chapter 8. IBM Tealeaf documentation and help 71

Appendix A. Cross-domain communication 73

Appendix B. Variations between jQuery and W3c flavors of CX UI Capture j2 . . 75

Notices 77
Trademarks 79
Privacy Policy Considerations 79

IBM Tealeaf CX UI Capture j2 Guide

For Release 8.6 and later, you must use the IBM Tealeaf CX UI Capture j2.

The IBM Tealeaf CX UI Capture j2 Guide details how to deploy the latest version of IBM Tealeaf UI Capture for your JavaScript-based web application. IBM Tealeaf UI Capture enables the monitoring of UI events in the visitor's browser that do not result in a request to the web application and cannot be captured through other means.

© Copyright IBM Corp. 1999, 2014

Chapter 1. IBM Tealeaf CX UI Capture j2

The IBM Tealeaf CX UI Capture j2 library enables the capture of page content, events, and other data that is not normally exchanged with the website.

By using the IBM Tealeaf CX Passive Capture Application, you can capture and replay your website with no modification to the site itself. However, if your site uses JavaScript to add interactivity, these interactions remain local to the web page. Actions taken that use these technologies may not be transmitted to the website and are therefore not captured by the CX Passive Capture Application.

By adding CX UI Capture j2 JavaScript to dynamic web pages, you can capture the state of the web pages and its JavaScript interactions. You can also capture other information only available in the page, such as page load times, the resolution of the user display, and other predefined attributes. You add a set of JavaScripts to relevant web pages, and the relevant content and actions are automatically captured and integrated with the rest of your IBM Tealeaf capture data.

The CX UI Capture j2 system obtains data from the browser and transmits it to the IBM Tealeaf capture server. This information includes mouse clicks, key presses, change events, and other information. The information is transmitted to the server by an asynchronous HTTP POST to a web page on the server that acknowledges the request. Because the request and response are captured by IBM Tealeaf, all the data is available to IBM Tealeaf for later reporting or replay.

Working with use cases

There are three basic uses for IBM Tealeaf CX UI Capture j2. You can also develop use cases specific to your needs.

These basic use cases are listed in order of increasing capability and complexity:

1. Capture browser environment information that is not typically transmitted to the web server. This data set includes:
   - Page render time and associated client-side performance timing measurements, as defined in the W3C Navigation Timing specification. See http://www.w3.org/TR/navigation-timing/.
   - The screen resolution of the client computer or device.

   More browser information can be captured through user agent information that is extracted and evaluated through IBM Tealeaf cxImpact. See "Managing User Agents" in the *IBM Tealeaf cxImpact Administration Manual*.
2. Capture predefined events on the page. IBM Tealeaf can capture your own events, which are described later.
3. Capture user interactions (UI) with the web page. These captures are typically the text that is typed into a form field, interaction with controls such as check boxes, select lists, or links.

You can develop your own use cases for IBM Tealeaf CX UI Capture j2. Specifically, you identify user activities, error conditions, and success criteria for application use. A robust set of use cases can help with the implementation process and in tracking the results in IBM Tealeaf.

© Copyright IBM Corp. 1999, 2014

Validating use cases with Client-Side Capture

IBM Tealeaf provides the Client-Side Capture utility, which enables the capture of web sessions to the local computer without access to an IBM Tealeaf CX UI Capture j2 capture deployment. Client-Side Capture (CSC) is useful for capturing use cases in the web application.

With CSC, web sessions can be captured and saved to your local computer for later replay in the stand-alone IBM Tealeaf CX RealiTea Viewer application. You can use CSC to visit all application areas and to capture specific workflows. You use this procedure to validate that IBM Tealeaf CX UI Capture j2 is properly capturing all requisite elements of the application.

Note: Use sessions that are captured by CSC to verify that all user interface events are being captured. If UI events are not being captured, check for the following issues.

- The web application is squashing these events. Review the application to verify that UI events are not being blocked or otherwise prevented from "bubbling" by the site code.
- The web application is dynamically updating content on the page. The IBM Tealeaf API must be called by the web application so that UI Capture can attach the appropriate event listeners.

See "Using Client-Side Capture for Fiddler" in the *IBM Tealeaf Client-Side Capture Manual*.

System prerequisites

Before you deploy IBM Tealeaf CX UI Capture j2, you make required modifications to your web application.

1. Statements to include the IBM Tealeaf JavaScript file must be added to those web pages of interest.
2. A target web page must be added to the origin web servers to acknowledge the POSTs being received from the IBM Tealeaf CX UI Capture j2 library on the browser.
3. Depending on your web page application, actionable user interface elements, such as form fields, buttons, and `<div>` tags, must be assigned unique HTML IDs to support later replay through IBM Tealeaf.
4. Any Ajax requests from JavaScripts running on the web page must have at least one unique data item so that multiple requests can be differentiated.

Supported browsers

The IBM Tealeaf CX UI Capture j2 JavaScript is supported for deployment to a specified set of browser applications and versions.

The supported web browsers follow.

- IE 6.0 and later
- Firefox 4.0 and later
- Safari 4.0 and later
- Opera 9.0 and later
- Chrome 10.0 and later

For replay, the IBM Tealeaf CX RealiTea Viewer uses an embedded instance of the version of Internet Explorer that is installed on the local computer.

Note: Replay of IBM Tealeaf sessions in the CX RealiTea Viewer, the stand-alone application for desktop replay, requires Internet Explorer support. If Internet Explorer is not supported by your enterprise's IT department, basic replay can be managed through the browser.

Note: Creation of events that monitor JSON-based data that is submitted from IBM Tealeaf CX UI Capture j2 is not supported in IBM Tealeaf CX RealiTea Viewer. You must use Browser-Based Replay to create events from these versions of CX UI Capture j2. See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.

For more information about these prerequisites, see Chapter 2, “CX UI Capture j2 installation and implementation,” on page 9.

- For a list of browsers and versions that are supported for use with the IBM Tealeaf Portal, see "CX Pre-Installation Checklist" in the *IBM Tealeaf CX Installation Manual*.

How CX UI Capture j2 for replay works

Of the three basic use cases for IBM Tealeaf CX UI Capture j2, capturing UI interactions with the web page for later replay is the most complex.

The following diagram shows this use case.

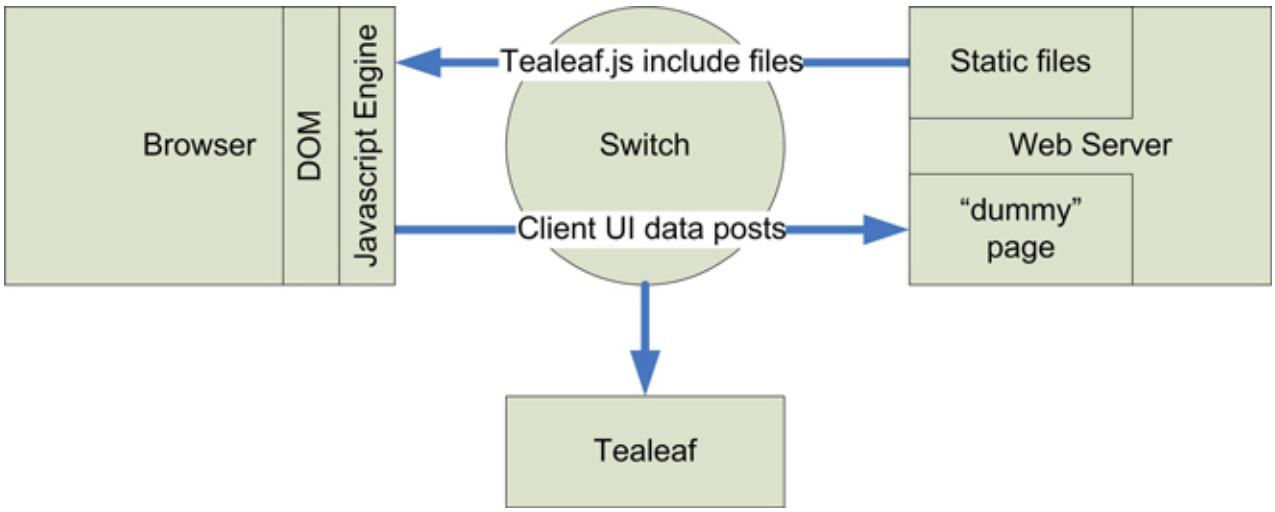


Figure 1. UI capture and replay

Suppose that you are attempting to capture a JavaScript, DHTML, or Ajax web page. Before the inclusion of the CX UI Capture j2 library, a visit to the page instructs the web server to return the page that contains the HTML/DHTML/JavaScript needed for the page functionality. As the user has interactions in the page, such as entering data on the form, the Ajax code on the page processes that data as designed.

- The web page can use Ajax techniques to post requests for data to the web server, which responds with the needed data. Thus, the page can be updated without causing a new page to load.

- When the user marks a check box, another Ajax interaction with the web server is triggered.
- The user can take an action that causes the current page to "unload", likely to be replaced by another web page.
  All these Request/Responses are captured by the IBM Tealeaf CX Passive Capture Application. During replay, IBM Tealeaf can often infer which forms were filled out and can bind the Ajax responses to Ajax requests made during replay.

However, without the CX UI Capture j2 library, IBM Tealeaf has no visibility into the interactions between the web page application and the visitor when the interaction did not trigger data transmission from the web server.

How CX UI Capture j2 captures interactions

After the CX UI Capture j2 library is added to the web page, the library code can detect UI events that occur on different elements in the page's Document Object Model (DOM).

CX UI Capture j2 can capture events from all frames or iframes that share the same domain as the parent page.

For example, when the user enters text into a field, the library code sees each key press event for the field and the actual key value, such as the letter `A`. The library stores these events into a temporary JSON data structure for eventual transmission back to the web server. The library can be configured to send this captured data up to the web server according to the following conditions:

- At predefined time intervals.
- When a predefined number of events were captured.
- At page unload time, usually defined as the minimum condition.

Each time the CX UI Capture j2 JavaScript software posts data to the web server, the IBM Tealeaf application on the web server acknowledges receipt of the data. This response is also captured by the IBM Tealeaf CX Passive Capture Application server, so the request/response pair can be accurately recorded.

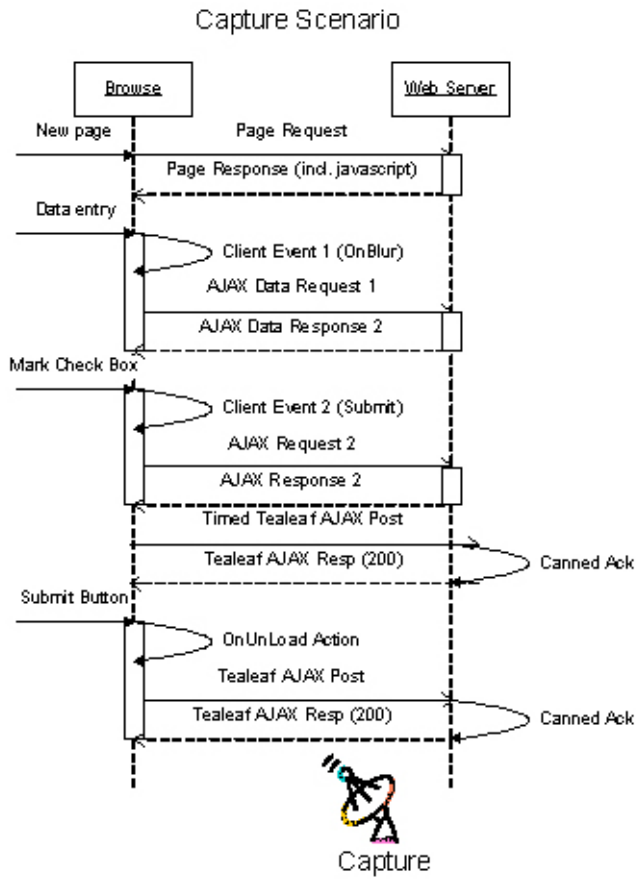


Figure 2. Ajax and CX UI Capture j2 transactions

This diagram shows the interactions of the example web page with the web server over time, including the web page Ajax transactions and the web pages added by CX UI Capture j2. In this sequence diagram, the web page application made two Ajax transactions followed by two IBM Tealeaf Ajax transactions. Time increases from top to bottom.

Note: For each user action, the CX UI Capture j2 solution submits a single message.

For more information about web page interactions, see Chapter 2, "CX UI Capture j2 installation and implementation," on page 9.

How CX UI Capture j2 replays captured interactions

To replay a web page, the IBM Tealeaf replay software mimics the interactions between the browser and visitor and between the browser and the web server.

These interactions are shown in the following diagram.

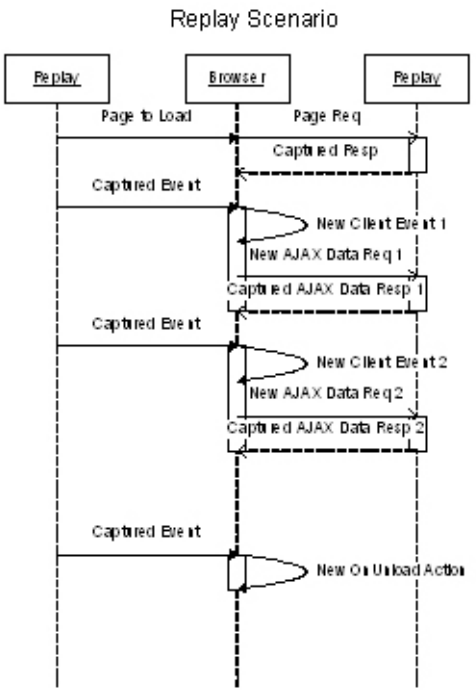


Figure 3. Replay scenario

The IBM Tealeaf replay software loads the captured page. When the page requests the CX UI Capture j2 JavaScript file, the replay software provides a different set of files, which are designed for replay.

The replay software then directly updates the state of the different objects on the page that is based on the visitor's actions. The IBM Tealeaf user can see these updates that occur at replay speed. Updates can cause the JavaScript on the page to run, which results in the Ajax POSTs being sent to the server. This request is trapped and mapped to the previously captured request and response. The latter is sent to the browser, so the JavaScript running on the page can do the same actions that occurred at capture time.

For replay to correctly map the new Ajax request to the previously captured one, a unique identifier for the post that is consistent between the captured instance and the replayed instance must be present. You can configure mapping rules so that request variables can be ignored or used to match to the captured request correctly.

Related documentation

A number of guides and resources are available to support implementation of the IBM Tealeaf CX UI Capture j2 solution.

Table 1. CX UI Capture j2 documentation resources

Document	Description
IBM Tealeaf Client Framework Data Integration Guide	After you deploy your client framework, more configuration can be required. Reference for configuring data capture in IBM Tealeaf and to make data available for creating events, which enables search and reporting. This document aids your IBM Tealeaf administrators whenever you are implementing an IBM Tealeaf client framework.
IBM Tealeaf CX Mobile Android Logging Framework Guide	Installation and implementation guide for the IBM Tealeaf CX Mobile Android Logging Framework for Android-based mobile native applications. Deployment requires the IBM Tealeaf CX Mobile license.
IBM Tealeaf CX Mobile iOS Logging Framework Guide	Installation and implementation guide for the IBM Tealeaf CX Mobile iOS Logging Framework for iOS-based mobile native applications. Deployment requires the IBM Tealeaf CX Mobile license.

Chapter 2. CX UI Capture j2 installation and implementation

Before you install and implement CX UI Capture j2, you must verify the software release that is used by your system.

Before you begin installation and implementation, review the guidelines for use. See Chapter 4, “CX UI Capture j2 usage guidelines,” on page 33.

Install version

Before you begin to implement CX UI Capture j2, verify that you have the latest version of the software.

You can access the installation package through IBM® Passport Advantage® Online.

This section contains details on how to implement of the IBM Tealeaf CX UI Capture j2 library.

Note: Depending on the complexity of your web application, an IBM Tealeaf CX UI Capture j2 library implementation can requireIBM consulting services. For more information, contact IBM Professional Services.

Support for CX RealiTea Viewer

The IBM Tealeaf CX RealiTea Viewer is a stand-alone desktop application that can be used to replay sessions captured by IBM Tealeaf, including client UI events.

To replay sessions that include UI events, you must install CX RealiTea Viewer on your local desktop.

Note: Replay of sessions that are captured by CX UI Capture j2 is not fully supported in the CX RealiTea Viewer.

- The JSON messages that contain UI events are not fully supported in the CX RealiTea Viewer.
- For replay of JSON messages, use Browser Based Replay. See "CX Browser Based Replay" in the *IBM Tealeaf cxImpact User Manual*.

Note: CX UI Capture j2 is compatible with Release 8.6 and later. It is not compatible with earlier releases.

Support for mobile web application capture

To capture data from mobile web applications, a license for IBM Tealeaf CX Mobile is required.

See "Client Framework Versions" in the *IBM Tealeaf Client Framework Data Integration Guide*.

CX Passive Capture Application configuration

By default, the IBM Tealeaf CX Passive Capture Application captures a number of the mimetypes that are posted by rich internet applications. Depending on the type of application you are deploying, you can enable the capture of more types.

© Copyright IBM Corp. 1999, 2014

IBM Tealeaf CX UI Capture j2 requires the CX Passive Capture Application to be configured to enable capture of JSON content in the request.

You must also configure the CX Passive Capture Application to capture the wanted POST traffic. CX Passive Capture Application can be configured to include or exclude specific file extensions, mimetypes, and POST types.

Note: Be sure to verify that your CX Passive Capture Application installation was configured to capture JSON mimetypes.

See "PCA Web Console - Pipeline Tab" in the *CX Passive Capture Application CX Passive Capture Application Manual*.

If dynamic JavaScript or CSS style sheets must be captured, the IBM Tealeaf CX Passive Capture Application must be configured to capture this content. Avoid capturing dynamic versions of content that is typically static. As an alternative, you can capture this content to a static archive. See "Managing Static Archives" in the *IBM Tealeaf cxImpact Administration Manual*.

- For descriptions of the capture types to enable for various types of RIA applications, see "Installation" in the *IBM Tealeaf CX Passive Capture Application Manual*.
- For more information about how to enable a capture type in the CX Passive Capture Application Web Console, see "PCA Web Console - Pipeline Tab" in the *IBM Tealeaf CX Passive Capture Application Manual*.

Installation and deployment plan

CX UI Capture j2 can be rapidly deployed in local environments for development and testing. When you resolve all issues that can be monitored outside of the IBM Tealeaf environment, the library can be ported to an IBM Tealeaf environment for further testing and tweaking.

Stage 1: Development environment

In the development environment, the goals for CX UI Capture j2 deployment are to integrate the required files into the web infrastructure and to establish basic connectivity between the web server, the client, and the CX Passive Capture Application for capture.

File to add to web pages

The CX UI Capture j2 JavaScript file must be included in each web page in the application that you want to monitor.

Server placement

On your web server, place the CX UI Capture j2 JavaScript file with the other static files served by your site.

Include location

Every served page that requires client event tracking must have `<script>` tags in it to load the CX UI Capture j2 JavaScript file.

In each page, the JavaScript file must be included as follows.

```
<script src="/tealeaf.js" type="text/javascript"></script>
```

If you place the IBM Tealeaf CX UI Capture j2 JavaScript file anywhere other than the root directory on your web server, you must adjust the `src=` parameter in the reference to point to the correct location.

Note: You are not required to place the CX UI Capture j2 JavaScript at the top of the page.

See Chapter 1, "IBM Tealeaf CX UI Capture j2," on page 1.

Installing CX UI Capture j2

To install and deploy CX UI Capture j2, complete the following steps.

1. If you did not do so already, configure the JavaScript.
2. Deploy the IBM Tealeaf target page in the location where the library is configured to POST to it. See "IBM Tealeaf target page" on page 17.
3. Place the required JavaScript in the appropriate web server location. See "Server placement" on page 10.
4. Modify the served pages to reference the JavaScript. See "Include location" on page 10.
5. When the JavaScript is referenced in the served pages and can post to the target page, CX UI Capture j2 is operational.

Verify Stage 1

To verify that CX UI Capture j2 is working properly in your Development environment, you complete the following tests.

No JavaScript errors on the page

Through your web browser, visit a sampling of the served pages. Verify that the included JavaScript is not generating JavaScript errors.

Verify client events are being posted

Deploy a browser monitoring tool such as Fiddler, HTTPWatch, or Firebug to verify that client events are being posted to the target page.

Note: Normal POSTs from CX UI Capture j2 are asynchronous, with one exception.

In the POST traffic stream, look for request headers that contain X-Tealeaf and a JSON message such as:

```
{"messageVersion":"2.1.0.0","serialNumber":1,"sessions":...
```

The version number varies from release to release.

Note: If UI Capture data is being forwarded to IBM Tealeaf, through the IBM Tealeaf Portal you can search for the IBM Tealeaf target page to see whether any sessions are retrieved. In the Testing and Production environments, use the Portal search test.

Stage 2: Testing environment

In the Testing environment, the goals are to deploy CX UI Capture j2 to verify correct operation and end-to-end monitoring of UI events, such that you can search for, replay, and report on client UI events.

Requirements for the Testing environment

Your Testing environment must meet certain requirements to complete end-to-end testing.

- IBM Tealeaf CX Passive Capture Application

  The CX Passive Capture Application must be implemented, enabled, and functioning properly. CX Passive Capture Application must be configured to capture the appropriate content types. See "Client Framework Versions" in the *IBM Tealeaf Client Framework Data Integration Guide*. For more information about configuration, see "PCA Web Console - Pipeline Tab" in the *IBM Tealeaf CX Passive Capture Application Manual*.

- Browser Based Replay

  To verify replay of UI events, use Browser Based Replay to verify that the UI events appear in the navigation pane. See “Replaying a captured session in BBR” on page 13.

- Search

  You must be able to search for and retrieve specific client UI fields in the request. Details follow.

  Verify basic search capability beforehand in both the Portal and CX RealiTea Viewer.

- Reporting

  Client UI events can appear in reports that are configured through the IBM Tealeaf Report Builder. See "Tealeaf Report Builder" in the *IBM Tealeaf cxImpact Reporting Guide*.

Installation in the Testing environment

To install and deploy in the Testing environment, follow the same steps that are used in the Development environment, modifying the JavaScript as needed to work in Testing.

- See “Installing CX UI Capture j2” on page 11.

Verify Stage 2

To validate your CX UI Capture j2 deployment in your Testing environment, complete the following tests.

Generate a session with UI events in it:

Using your web browser, visit the web application that is monitored by the IBM Tealeaf Testing environment.

Requirements

- Do not use Client-Side Capture for this test. Use IBM Tealeaf to capture and process the data.
- Visit a series of pages that contain UI events. Trigger all test UI events so that they are included in your captured session.
- Generate the session so that you have a uniquely identifiable way to locate the session through search.
  - As the session is being generated, you can use Fiddler or a similar tool to review the request data to locate the IBM Tealeaf session identifier.
  - You can also do browsing tricks such as inserting a unique string in a form field and then submitting it.
- Be sure to close your session so that it can be indexed for search.

Replaying a captured session in BBR:

After you capture a session, you replay it through Browser Based Replay to confirm that client UI events are present. Based on your review of the replay, you can adjust configuration settings and develop replay rules, after which you can replay the session.

Repeat this process until you secure a good quality replay of the captured session.

Note: Replay of captured sessions in CX RealiTea Viewer is not fully supported for CX UI Capture j2, depending on your application. Beginning in Release 8.6 or later, you use Browser Based Replay for replay.

1. Locate the session through search. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
2. Replay the session in Browser Based Replay.
3. Switch to request view.
4. Step through the pages of the session.
5. At the top of the display panel, the following link displays.
   `Click here to view Step Attributes`
6. This link indicates that JSON data was captured as part of the session, which means that CX UI Capture j2 is submitting it correctly.
   - You can click the link to review the JSON data in a readable form.
   - From this area, you can create IBM Tealeaf events and step attributes of the data. See “Create IBM Tealeaf events for client UI data.”
   - When the IBM Tealeaf events are created from the JSON data, those events are marked in subsequent sessions that are captured by IBM Tealeaf. Then, you can search for them using the Portal interface. See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
7. To ensure a good replay experience, you can create replay rules or modify your existing ones to properly manage the replay of sessions that are captured by IBM Tealeaf CX UI Capture j2. See “Tweak replay rules” on page 14.
8. After IBM Tealeaf events are created, you can create reports in the IBM Tealeaf Report Builder with these events to begin tracking client-side events. See “Verify IBM Tealeaf Portal reporting” on page 14.

Create IBM Tealeaf events for client UI data:

After you implement CX UI Capture j2, you can test for the availability of data for eventing purposes through Browser Based Replay.

Browser Based Replay enables the creation of attributes and events from the JSON steps that are submitted from the IBM Tealeaf client frameworks. Using this simple mechanism, you can quickly create the step attributes and events that are needed to capture traffic from the client interface, as submitted by CX UI Capture j2.

See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.

Note: If you are upgrading from a UI Capture version from 2012.06.01.1 or earlier, the following request fields are not populated by this version of UI Capture:

- `TLT_CUI_URL`
- `TLT_CUI_APPLICATION_NAME`

Any eventing, indexing, or search templates configured based on the presence of these request fields do not work for CX UI Capture j2.

- For more information about events, see "TEM Events Tab" in the *IBM Tealeaf Event Manager Manual*.
- For more information about indexing, see "Configuring CX Indexing" in the *IBM Tealeaf CX Configuration Manual*.
- For more information about configuring search templates, see "Configuring Search Templates" in the *IBM Tealeaf cxImpact Administration Manual*.

Tweak replay rules:

Client UI events often require replay rules to manage proper display of them. A replay rule is an action that is applied to session data before replay for purposes of enhancing the replay experience.

Note: Replay rules are stored in a user profile. When CX UI Capture j2 is deployed in a Production environment, you make this profile available to all CX RealiTea Viewer and Browser Based Replay users. See “Deploy profile.”

Replay rules can be applied through Browser Based Replay. See "BBR Replay Rules" in the *IBM Tealeaf cxImpact User Manual*.

Repeat the tests:

Repeat the tests in this section until you achieve a satisfactory replay.

To iterate on the process, see “Replaying a captured session in BBR” on page 13.

Verify IBM Tealeaf Portal reporting:

You can review the following reports to verify that client UI events are being posted to IBM Tealeaf and submitted to the reporting databases.

After you create an IBM Tealeaf event with Step-Based Eventing in Browser Based Replay, wait at least 1 hour for report data to gather.

Then, you can create a simple report in the IBM Tealeaf Report Builder containing your event and constrained to the current date as the focus period. If data is present, then the reporting functions are working for CX UI Capture j2 data.

See "Tealeaf Report Builder" in the *IBM Tealeaf cxImpact Reporting Guide*.

Stage 3: Production environment

After you complete all of the tests in your Testing environment, you can deploy the CX UI Capture j2 solution to your Production environment.

Deploy profile

If you develop profile rules or modify CX RealiTea Viewer settings in your Testing environment, make the changes available to all CX RealiTea Viewer and Browser Based Replay users who can access the Production environment.

- For more information about sharing your CX RealiTea Viewer profile, see "RealiTea Viewer - Profile Options" in the *IBM Tealeaf CX RealiTea Viewer User Manual*.

- For Browser Based Replay, replay profile settings are configured for individual user groups for IBM Tealeaf cxImpact or IBM Tealeaf cxReveal users.
  - For more information, see the Browser Replay Profile in "CX User Administration" in the *IBM Tealeaf cxImpact Administration Manual*.
  - For more information, see the Browser Replay Profile in "cxReveal User Administration" in the *IBM Tealeaf cxReveal Administration Manual*.

Install in Production

To install and deploy in the Production environment, follow the same steps that are used in the Development environment, modifying the JavaScript as needed to work in Production.

- See “Installing CX UI Capture j2” on page 11.

Verify Stage 3

Repeat all tests that you completed in Development and Testing in the Production environment.

- “Verify Stage 1” on page 11
- “Verify Stage 2” on page 12

Configuring the JavaScript

CX UI Capture j2 JavaScript interactions with web application pages

The IBM Tealeaf CX UI Capture j2 JavaScript is able to monitor user interface events by attaching keyboard and mouse event listeners to HTML elements on the pages of your web application.

Overriding existing API

In some cases, IBM Tealeaf overrides the existing browser API.

Currently, this override behavior occurs for the following event.

window.onerror

IBM Tealeaf overrides the default `window.onerror` event handler to track client-side error conditions that do not generate a transaction with the web server. The IBM Tealeaf CX UI Capture j2 library performs this override only if there is no current listener for `window.onerror`.

Block sensitive data fields

If required, sensitive information that is submitted by visitors into client-side forms can be blocked in the JSON data that is submitted to IBM Tealeaf.

For example, if a visitor enters a social security number into a form field, you can configure field block rules to block the entered social security number (such as `XXX-XX-XXXX`).

The following section describes how privacy works in IBM Tealeaf CX UI Capture j2. You implement your privacy configuration as part of your initial installation. For more information, contact IBM Professional Services.

Installation on web pages

When you plan the installation of the CX UI Capture j2 library, consider that it does use system resources to process on the website and the IBM Tealeaf capture systems. Install the library only on the pages that you want to capture and replay.

Installing the CX UI Capture j2 library can cause the following effects.

- Page load times can increase slightly during first download. IBM Tealeaf provides strategies for minimizing the effect on load times, including caching the JavaScript after initial download.
- More data is posted by the web page to the server, but it is not processed.
- IBM Tealeaf must process and store the additional data.

IBM Tealeaf CX UI Capture j2 is designed to minimize these burdens.

Unique HTML IDs

To capture the client events for a Document Object Model object, each object or control available with which the visitor can interact must have a unique HTML ID. Without unique IDs, IBM Tealeaf cannot properly replay the UI events in the page.

If you did not instrument your application to support unique IDs or unique names in the document object model, IBM Tealeaf CX UI Capture j2 automatically generates paths to the node that support the requirement of uniqueness.

IBM Tealeaf also supports the creation of ID black lists, which forces the use of Xpath or name attribute values as identifiers. See Chapter 5, “CX UI Capture j2 reference,” on page 41.

Cookies

The web session must use cookies for sessionization. Cookies allow the client events to send messages without having to understand the sessionization mechanism.

For best results in capturing client UI events, deploy the IBM Tealeaf Cookie Injector in your web server environment. The IBM Tealeaf Cookie Injector provides unique session IDs regarding captured IBM Tealeaf session data. See "Installing and Configuring the Tealeaf Cookie Injector" in the *IBM Tealeaf CX Cookie Injector Manual*.

Installation on the web server

The IBM Tealeaf CX UI Capture j2 library consists of a single JavaScript file. You place this JavaScript file with the other static files served by your site.

References to the JavaScript file

To track activity on a web page, you add a reference to the required CX UI Capture j2 JavaScript file.

Every served page that requires client event tracking must have `<script>` tags in it to load the IBM Tealeaf CX UI Capture j2 JavaScript:

```
<script src="/Tealeaf.js" type="text/JavaScript"></script>
```

If you place the CX UI Capture j2 JavaScript in a place other than the root directory on your web server, adjust the `src=` parameter in the reference to point to the correct location.

See "UI Capture j2 Overview".

IBM Tealeaf target page

The IBM Tealeaf target page is a URL to which the CX UI Capture j2 JavaScript is configured to send POST requests for capture by the IBM Tealeaf system. You place this dynamic page with the other dynamic page files for your web application.

This dynamic target web page must be posted or created to receive the information posted by the JavaScript, such as `TealeafTarget.php`, `TealeafTarget.jsp`, `TealeafTarget.asp`, or similar.

Note: The target page exists only to acknowledge the post. No data is saved.

The web server returns a minimal valid response, such as the following message.

```
<html><head></head><body>Received 895 bytes in 0.1 ms</body></html>
```

Place the dynamic target page with your other dynamic page files. For simplicity, you can place these files in the root directory. Your website administrator can provide you with the policies and procedures for your installation.

Sample targets for ASPX, JSP, and PHP environments are provided with the CX UI Capture j2 distribution.

- Modify the target page for your specific target environment.
- Use these sample targets as the basis for creating target pages suitable for other environments.

Verifying the target page

All of the provided IBM Tealeaf target pages are named so that the data they receive can be captured, processed, and integrated into replay by IBM Tealeaf.

Note: The file name of the IBM Tealeaf target page must include `TealeafTarget` in it for easy identification. Do not change this file name after you configure IBM Tealeaf CX UI Capture j2.

Complete the following steps to verify that there are no impediments to the capture and processing of the IBM Tealeaf target page in your system.

1. In the Web Console of the IBM Tealeaf CX Passive Capture Application, you configure file extensions to drop from capture. Verify that the file extension used for your IBM Tealeaf target page is not dropped from capture. See "PCA Web Console - Pipeline Tab" in the *IBM Tealeaf CX Passive Capture Application Manual*.
2. In the IBM Tealeaf CX RealiTea Viewer, some file extensions are treated as interpreted pages, which means that the pages are interpreted on the web server before they are delivered to CX RealiTea Viewer. Verify that the extension for your IBM Tealeaf target page file name does not appear in the list of possible interpreted pages in CX RealiTea Viewer. See "RealiTea Viewer - Advanced Options Tabs" in the *IBM Tealeaf CX RealiTea Viewer User Manual*.

Target page installation

The target page must be added to your web infrastructure in a location that can be accessed from the visitor's web browser. For simplicity, you can place these files in the root directory or in a central JavaScript directory.

Your website administrator can provide you with the policies and procedures for your installation.

In your web application's pages, references to the target page must be relative to the site root path. For `www.example.com`, to reference an IBM Tealeaf target page at `www.example.com/scripts/TealeafTarget.aspx`, you configure the IBM Tealeaf CX UI Capture j2 to `/scripts/TealeafTarget.aspx`. The initial / makes it a site-root relative path.

Deployment guidelines

Refer to these guidelines when you deploy the IBM Tealeaf target page.

Typically, the IBM Tealeaf target page is deployed independently of the core application itself in logical and physical terms. Review these guidelines with your infrastructure staff.

- Avoid deploying the IBM Tealeaf target page behind enterprise security features, such as SiteMinder.
- If possible, deploy the IBM Tealeaf target page on a separate application server instance from the web application. While it is a lightweight deployment, it can compete for resources with the web application. If possible, deploy it on a dedicated application server instance.
- Test the IBM Tealeaf target page thoroughly in a test environment before deployment to production.

References to the target page

In your web application's pages, references to the IBM Tealeaf target page must be relative to the site root path.

For `www.example.com`, to reference an IBM Tealeaf target page at `www.example.com/scripts/TealeafTarget.aspx`, you configure the CX UI Capture j2 library to POST to `/scripts/TealeafTarget.aspx`. The initial "/" makes it a site-root relative path.

In the CX UI Capture j2 JavaScript, the URL of the target page is configured as part of the initial implementation.

Unit tests of the target page

To test that the target page is operating properly, you can run the following tests.

- GET: Enter the URL of the target page in your browser. If you receive a blank page or non-404 error, the page is properly handling your request.
- POST: You can use Fiddler or another traffic monitoring tool to generate a post to the page. Verify that the POST action resulted in a Status Code 200 message.

Configuration of CX UI Capture j2 JavaScript

To configure the JavaScript file for IBM Tealeaf CX UI Capture j2, you define settings for compression and caching.

Compression

For optimal performance, compress the IBM Tealeaf JavaScript on your web server.

- Configure the web server to perform compression encoding of the response data before it is sent back to the browser, which decompresses it for use on the client.
- Apply compression encoding to static text content, including the IBM Tealeaf JavaScript file.

  Note: Since this behavior is negotiated between server and client, an HTTP 1.1-compliant web server must not use any compression encoding unless the client indicated support of it by a Request Header (`Accept-Encoding`).

This form of compression further reduces the file sizes of minified IBM Tealeaf JavaScript by approximately 75% before delivery to the client.

Browser caching

For browser caching configuration, follow these guidelines.

- Set the TTL values for the `Expires` or `Cache-Control` headers for the IBM Tealeaf JavaScript to the same value as the rest of the scripts that are stored on the web server.
- Take advantage of the conditional GET feature of HTTP 1.1 by configuring your web server to set the appropriate `ETag` or `Last-Modified` headers. When this feature is enabled, any change to the JavaScript resources is automatically propagated to the visitor's cache.

For more information, see these industry resources.

- `http://developer.yahoo.com/performance/rules.html`
- `http://code.google.com/speed/page-speed/docs/rules_intro.html`

IIS configuration

On Windows Server 2003, put the IBM Tealeaf CX UI Capture j2 files in a directory with the other JavaScript file to comply with the "Script source access" setting in the configuration.

For more information about configuring IIS, see http://support.microsoft.com/default.aspx?scid=kb;en-us;313075.

Non-IIS web server configuration

For a non-IIS server, a dynamic web page must be created to accept the POST from the IBM Tealeaf CX UI Capture j2 JavaScript. The contents of the web page vary based on the deployed technology.

The contents of the returned UI Capture JavaScript POST requests are irrelevant. To minimize bandwidth, minimize the return contents.

Web page modifications

Every served page that requires client event tracking must have a `<script>` tag in it to load the IBM Tealeaf CX UI Capture j2 JavaScript file.

See “Include location” on page 10.

Change management for Document Object Model elements

In certain instances, when your web application changes the Document Object Model, your page code must notify IBM Tealeaf of the location where the changes were made.

For Ajax, DHTML, or JavaScript based updates that add or edit Document Object Model elements, the page in which the changes are made must call the following IBM Tealeaf API to register the node change, `TLT.rebind`.

Note: This call must be made when new nodes are added for which capturing the user interaction is essential. Changes that remove nodes can be processed during normal cleanup when the page is unloaded.

Example code:

```
<html>
<head>
<title>Rebind example</title>
<script type="text/javascript">

...
/* AJAXUpdate will be called after the XMLHttpRequest returns
 * a successful response.
 */
function AJAXUpdate(responseMarkup){
  var targetDiv = document.getElementById("target_div");
  if (!targetDiv || !responseMarkup) {
    return;
  }
  targetDiv.innerHtml = responseMarkup;
  if (TLT && TLT.rebind) {
    // Notify Tealeaf of the DOM update.
    TLT.rebind(targetDiv);
  }
}

...

</script>
</head>
<body>

...

  <div id="target_div">Placeholder where the AJAX response markup will
      be placed.
  </div>

...

</body>
</html>
```

Upgrade CX UI Capture j2

For information about upgrading to the latest version of IBM Tealeaf CX UI Capture j2, contact IBM Professional Services.

Support for legacy headers

For customers that are upgrading from IBM Tealeaf CX UI Capture for AJAX to IBM Tealeaf CX UI Capture j2, support for some of the legacy headers submitted from CX UI Capture for AJAX is retained.

Header
Description

X-TeaLeafType
Type of client-side event set. Default value is `GUI`.

X-TeaLeaf-Page-Url
Full URL of the parent page.

X-TeaLeaf-Page-Render
Time in milliseconds for page to render.

X-TeaLeaf-Page-Objects
Number of objects (object tags) on page.

X-TeaLeaf-Page-Img-Fail
Number of bad images on the page.

X-TeaLeaf-Page-Cui-Exceptions
Number of exceptions on the page.

X-TeaLeaf-Page-Dwell
Dwell time on the page.

X-TeaLeaf-Page-Last-Field
ID of the last visited form field on the page.

X-TeaLeaf-Visit-Order
Visit order of the form fields on the page.

Note: Other headers that are submitted through IBM Tealeaf CX UI Capture for AJAX are not supported. For more information about those headers, see "UI Capture for Ajax Sample Client Event Message" in the *IBM Tealeaf CX UI Capture for AJAX Guide*.

Uninstalling CX UI Capture j2

To uninstall CX UI Capture j2, apply the following changes in your Development environment.

1. Save a copy of the currently deployed CX UI Capture j2 JavaScript. If you minified the JavaScript, save a version of the source JavaScript.
2. Remove the `<include>` references to the CX UI Capture j2 files from the pages of your web application.
3. Remove the CX UI Capture j2 files from the area where they are installed on your web server.
4. Remove the IBM Tealeaf target page.
5. Push the changes to the server.
6. Verify that no browser errors are generated in visitor browsers when you visit your own web application.
7. Verify that IBM Tealeaf CX UI Capture j2 events are no longer being captured and processed by IBM Tealeaf.
8. Disable any IBM Tealeaf event objects that reference UI events captured by IBM Tealeaf CX UI Capture j2.

If the changes are applied successfully in your development environment, you can apply them in your Test and Production environments in succession.

Chapter 3. CX UI Capture j2 Configuration Wizard

With the Configuration Wizard, you can complete the necessary configuration tasks to modify your IBM Tealeaf CX UI Capture j2 solution to work with your web application. Through its step-based interface, you configure IBM Tealeaf CX UI Capture j2, after which the configuration changes can be applied to the JavaScript in your environment.

Use the configuration wizard only to perform a basic configuration of CX UI Capture j2. The configuration wizard is only intended as a starting point, and does not provide detailed configuration options to better optimize the installation. For full scale implementations, consult IBM Professional Services or a knowledgeable web developer.

The Configuration Wizard is available in the installation package in the `Wizard` directory.

In the Wizard, you can access context-specific documentation through the **Help** (

) icon next to available options.

Note: For Release 8.6 and later, you must use CX UI Capture j2, the new version of the UI Capture SDK.

Supported browsers

The IBM Tealeaf CX UI Capture j2 Configuration Wizard is supported in Chrome and Firefox 10 and later.

Internet Explorer is not supported.

Configuration Wizard controls

When you use the CX UI Capture j2 Configuration Wizard, you complete different actions with these user interface controls.

- To proceed to the next screen in the wizard, click **Next**.
- To return to the previous screen, click **Previous**.
- To complete the configuration, click **Finish**.
- To reset the configuration to the defaults provided by Tealeaf, click **Reset to defaults**.
- To define and test regular expressions for use in your configuration, click **RegEx tester**. See “RegEx Tester” on page 30.
- For more information about individual configuration settings, click the **Help** (

) icon next to the setting.

Build types

By default, the Configuration Wizard generates the IBM Tealeaf CX UI Capture j2 JavaScript for use in a Production environment. Depending on your current state of development, you can select a different build type in the Configuration Wizard.

© Copyright IBM Corp. 1999, 2014

Production versions of the JavaScript are minified, which means:

- Comments and other non-functional text is removed.
- Variable names are reduced to the smallest possible files.
- Overall JavaScript payload is reduced to ease bandwidth requirements.
- Code is no longer easily read.

Descriptions of the other build types follow.

Build Type
Description

`Production build (minified)`
The default setting.

`Production build (non-minified)`
All production build options are applied. However, the JavaScript is not minified, which enables review of the deployed code.

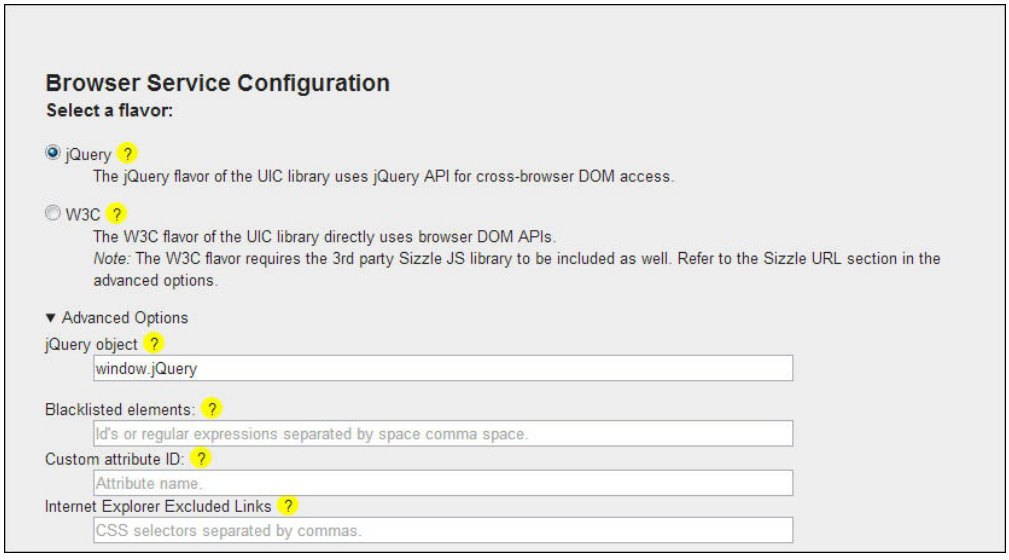
`Development build (non-minified)`
Extra debug code is enabled.

Note: This option is primarily for developers or power users of the library who want to debug the code. Do not deploy this type of build in production.

Note: If you are still developing and testing your IBM Tealeaf CX UI Capture j2 solution, select the `Production build (non-minified)` build option.

Step 1: Browser service configuration

In this step, you configure the Browser Service, which is used to monitor objects in the browser.


Browser Service Configuration
Select a flavor:
jQuery
The jQuery flavor of the UIC library uses jQuery API for cross-browser DOM access.
W3C
The W3C flavor of the UIC library directly uses browser DOM APIs.
Note: The W3C flavor requires the 3rd party Sizzle JS library to be included as well. Refer to the Sizzle URL section in the advanced options.
▼ Advanced Options
jQuery object
window.jQuery
Blacklisted elements:
Id's or regular expressions separated by space comma space.
Custom attribute ID:
Attribute name.
Internet Explorer Excluded Links
CSS selectors separated by commas.

You choose either jQuery or W3C. Depending on what you select, you have different options under the Advanced Options section.

If you select jQuery, the following Advanced Options are available.

- jQuery object: Add a path to the jQuery object.

- Blacklist elements: Blacklist any element ids that are not unique or dynamically generated. Element IDs that match with any of the blacklisted entries are replaced with custom attribute values or XPATH.
- Custom attribute ID: Add one or more attribute names that can be used to uniquely identify an element when its HTML id is not available or blacklisted.
- Internet Explorer Excluded Links: This configuration is specified as an array of CSS selectors. For example, the configuration would be specified as `ieExcludedLinks: ['a.ignore'],` to ignore the beforeunload that is triggered by the link, `< a href ='javascript:process();' class='ignore'>Click< /a>`.

If you select W3C, the following Advanced Options are available.

Note: When using W3C, if you have iFrame on your website, any actions within the iFrame on a Legacy IE browser does not bubble up to be captured currently.

- Sizzle URL: Add the Sizzle URL. Sizzle is required for the correct operation of the library in legacy IE browsers when the W3C service is used.
- Sizzle object: Include the path to the Sizzle object.
- Blacklisted elements: Blacklist any element ids that are not unique or dynamically generated. Element ids that match with any of the blacklisted entries are replaced with custom attribute values or XPATH.
- Custom attribute ID: Add one or more attribute names that can be used to uniquely identify an element when its HTML id is not available or blacklisted.
- Internet Explorer Excluded Links: This configuration is specified as an array of CSS selectors. For example, the configuration would be specified as `ieExcludedLinks: ['a.ignore'],` to ignore the beforeunload that is triggered by the link, `< a href ='javascript:process();' class='ignore'>Click< /a>`.

For the Internet Explorer Excluded Links option, a `CSS selector` value results in an exception in Chrome and Webkit browsers, if an invalid character (for example $) is specified, and it is not properly escaped with \.

Example:

```
{
//Name Value Blocking
"targets": ["input[name=login\\$password]"],
"maskType" : 2 //Replace with XXX's
}
```

Click in the Configuration Wizard for more details on each setting.

More details are available in the Reference section. See Chapter 5, “CX UI Capture j2 reference,” on page 41.

Step 2: Queue service configuration

In this step, you configure the message queue that is used to store JSON messages locally before they are submitted to the IBM Tealeaf target page for capture and processing.

You can configure only one queue. This queue must be named `DEFAULT`.

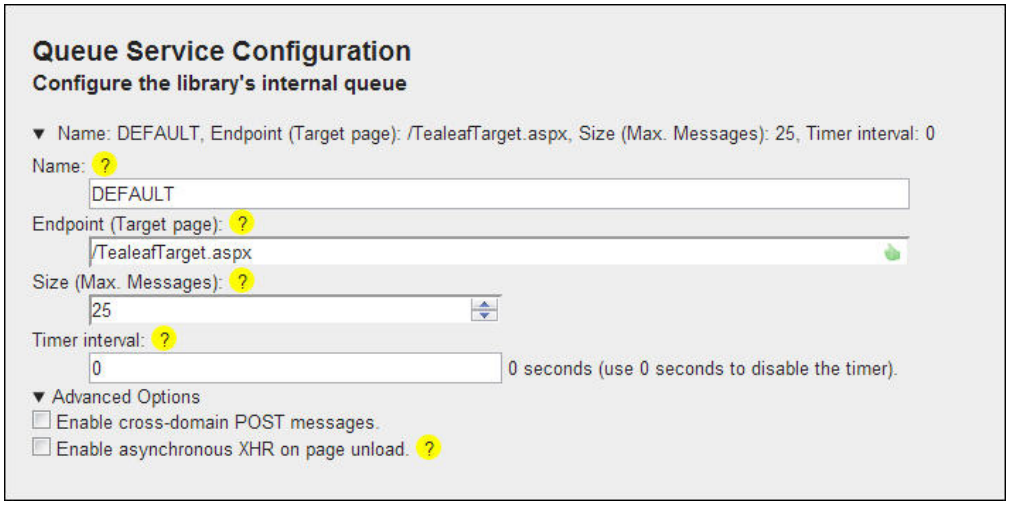
Update the `Endpoint` field with the target page URL.

Update the `Size` field with the limit to the number of messages to permit in the queue before it is sent. By default, messages are queued locally in the visitor's browser. When the queue reaches a defined number of messages, the data that is contained in it is serialized and submitted to the IBM Tealeaf target page, which enables IBM Tealeaf to capture and process the messages.

For enabling shadow browsing scenarios, you can set the `Timer Interval` to periodically flush the queue irrespective of the number of messages. In most other cases, it is best to leave this setting disabled.

You can select `Enable cross-domain POST messages` and `Enable asynchronous XHR on page unload` to enable these options.

Note: Enabling asynchronous request on page unload might result in incomplete or missing data.



Click in the Configuration Wizard for more details on each setting.

More details are available in the Reference section. See Chapter 5, “CX UI Capture j2 reference,” on page 41.

Step 3: Message service configuration (privacy masking configuration)

In this step, you can configure privacy settings so that sensitive data detected on the client is blocked or masked before submission to IBM Tealeaf for capture.

Configuration of privacy through IBM Tealeaf CX UI Capture j2 ensures that sensitive data never touches IBM Tealeaf.

IBM Tealeaf provides multiple methods for applying privacy throughout the IBM Tealeaf solution. See "Managing Data Privacy in Tealeaf® CX" in the *IBM Tealeaf CX Installation Manual*.

In the Configuration Wizard, you can configure multiple privacy masks. Each privacy mask configuration consists of at least one Target, which includes an `ID` , `IDTypes`, and `CSS Selector`. Each privacy mask configuration also consists of a `MaskType`. To remove an individual target from a privacy mask configuration, click **Remove Target**. To remove an entire privacy mask configuration, click **Remove Privacy Configuration**.

Empty privacy masks that you add in the Wizard are not added to the configuration file.

The value that you enter for the `CSS selector` results in an exception in Chrome and Webkit browsers, if an invalid character (for example $) is specified, and it is not properly escaped with \.

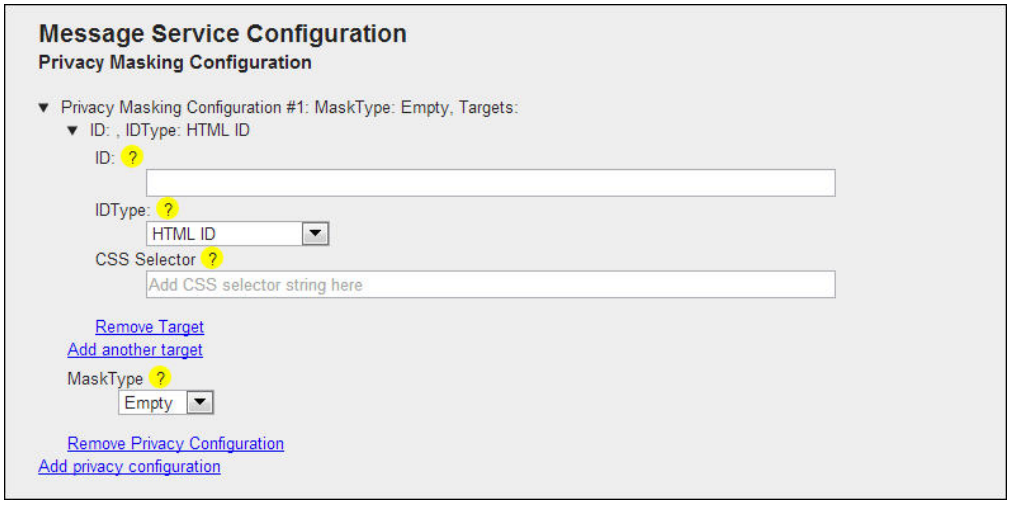
Example:

```
{
//Name Value Blocking
"targets": ["input[name=login\\$password]"],
"maskType" : 2 //Replace with XXX's
}
```

Password fields can be masked if you use the `defaultconfiguration.js` and set the `input type=password fields`. If you use the Configuration Wizard, then the password fields must be masked by explicitly adding the following to the privacy configuration.

```
{
    targets: [
        // CSS Selector: All password input fields
        "input[type=password]"
    ],
    "maskType": 3
}
```

More details are available in the Reference section. See Chapter 5, “CX UI Capture j2 reference,” on page 41.



Click in the Configuration Wizard for more details on each setting.

Step 4: Serializer

In this step, you can elect to use a built-in parser/serializer if none is available.

Add parsers that contain functions for CX UI Capture j2 to use (for example, `JSON.parse`). The first parser is the most important. If CX UI Capture j2 does not find the first parser, CX UI Capture j2 tries again, and so on.

Add serializers that contain functions for CX UI Capture j2 to use (for example, `JSON.stringify`). The first serializer is the most important. If CX UI Capture j2 does not find the first serializer, CX UI Capture j2 tries again, and so on.

Click in the Configuration Wizard for more details on each setting.

More details are available in the Reference section. See Chapter 5, "CX UI Capture j2 reference," on page 41.

Step 5: Modules

In this step, you select the modules to enable in your IBM Tealeaf CX UI Capture j2 solution, and set advanced options for each module.

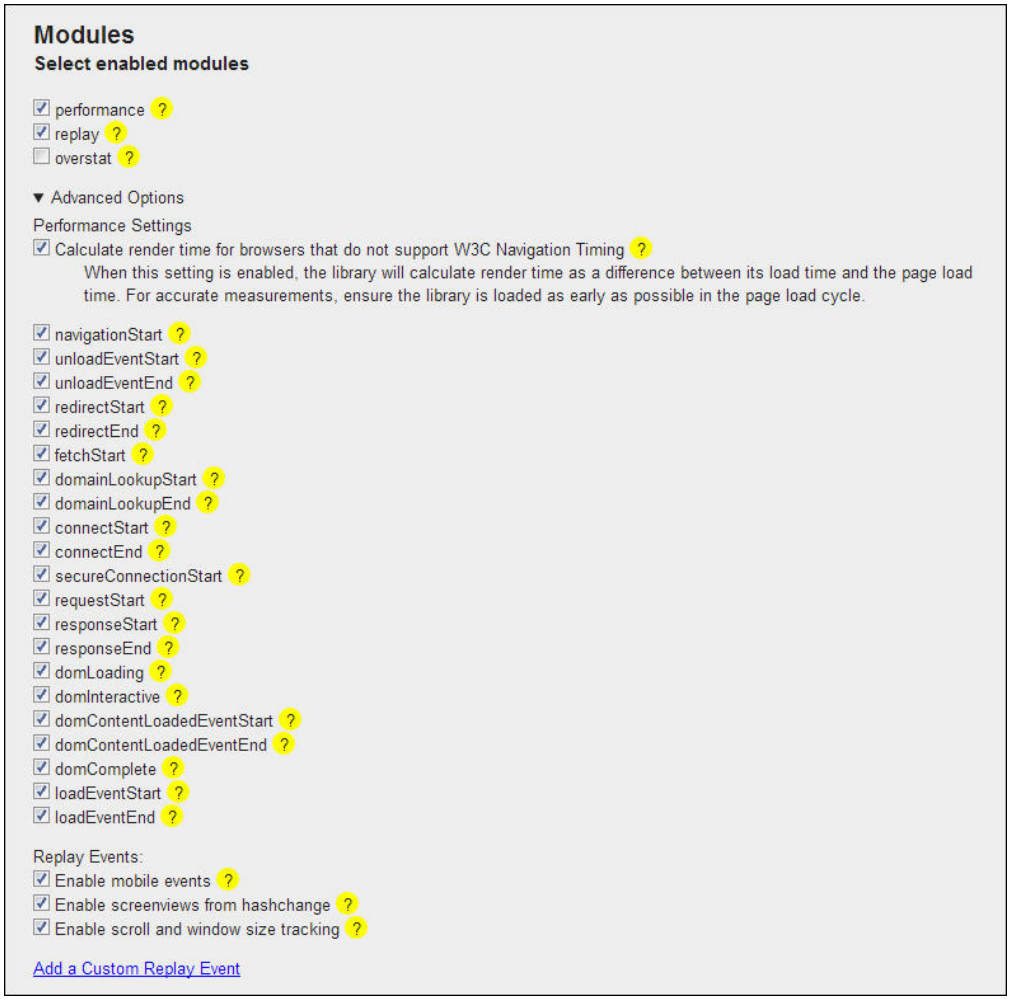
You can also add custom replay events.

When adding a custom replay event, a `CSS selector` value results in an exception in Chrome and Webkit browsers, if an invalid character (for example $) is specified, and it is not properly escaped with \.

Example:

```
{
//Name Value Blocking
"targets": ["input[name=login\\$password]"],
"maskType" : 2 //Replace with XXX's
}
```

The Performance module events comply with the W3C standard on navigation timing. The performance module parses the W3C Navigation Timing performance object and includes this information as the performance message in the JSON data stream sent back to the Tealeaf capture server.



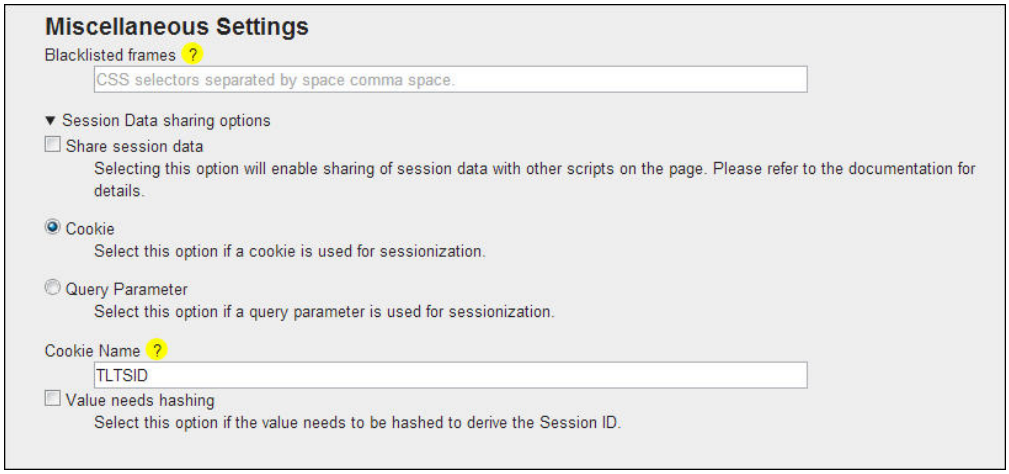
Click in the Configuration Wizard for more details on each setting.

More details are available in the Reference section. See Chapter 5, “CX UI Capture j2 reference,” on page 41.

For more information, visit http://www.w3c.org.

Step 6: Miscellaneous settings

In this step, you can add Blacklisted frames that are excluded from data collection. You can also set Session Data sharing options.


Miscellaneous Settings
Blacklisted frames
CSS selectors separated by space comma space.
Session Data sharing options
Share session data
Selecting this option will enable sharing of session data with other scripts on the page. Please refer to the documentation for details.
Cookie
Select this option if a cookie is used for sessionization.
Query Parameter
Select this option if a query parameter is used for sessionization.
Cookie Name
TLTSID
Value needs hashing
Select this option if the value needs to be hashed to derive the Session ID.

For the Blacklisted Frames option, a `CSS selector` value results in an exception in Chrome and Webkit browsers, if an invalid character (for example $) is specified, and it is not properly escaped with \.

Example:

```
{
//Name Value Blocking
"targets": ["input[name=login\\$password]"],
"maskType" : 2 //Replace with XXX's
}
```

Click in the Configuration Wizard for more details on each setting.

More details are available in the Reference section. See Chapter 5, “CX UI Capture j2 reference,” on page 41.

RegEx Tester

In the RegEx Tester, you can specify regular expressions and test them against data that you provide.

Figure 4. RegEx Tester

1. **RegEx**: Insert your regular expression to test here.
   - By default, searches in the **Test Sample** data are case-sensitive. You can configure the test to do **Case insensitive** searches.
   - By default, the regular expression is matched against the first occurrence in the string.
   - To match the regular expression against all occurrences in the string, select the **Global**. This option is not typically required.
2. **Test Sample**: Insert your sample data to apply in the test here. Use data specific to your web application, such as a sample request.
3. **Matches**: If a match is found for the regular expression in the test sample data, this value is `true`.
4. **RegEx (ready)**: Whether the regular expression evaluates to `true` or `false`, the value that you can use to insert the regular expression into any Configuration Wizard text box displays here. Copy and paste it as needed.
   - References to the case-insensitive or global flags are inserted. For example, the case-insensitive flag is added to the expression in the following manner: `"flags": "i"`.
   - For more information about regular expression flags, see https://developer.mozilla.org/en- US/docs/JavaScript/Reference/Global_Objects/RegExp.
5. Click **Test**.
6. To close the RegEx Tester, click ✖.

Chapter 4. CX UI Capture j2 usage guidelines

This chapter provides guidelines for successful implementation and use of IBM Tealeaf CX UI Capture j2.

In parallel with your process for developing the web application, plan to deploy IBM Tealeaf CX UI Capture j2 as early as possible in the development process for the following reasons.

- Early deployment limits the potential for issues that are discovered in a production environment.
- Early deployment provides an opportunity to monitor application problems and help debug the application.

These guidelines apply to all deployments of IBM Tealeaf CX UI Capture j2. Different options can apply to different web application types.

Application scope

IBM Tealeaf CX UI Capture j2 supports specific browsers, application types, and protocols.

For a list of supported browsers, see Supported Browsers.

Supported application types

In general, rich internet application (RIA) functionality must be limited to only the areas of your web application that require it.

RIA requires more sophisticated monitoring techniques and can increase the volume of traffic that IBM Tealeaf must process.

IBM Tealeaf supports the following rich internet application types:

- Ajax

Note: IBM Tealeaf continues to support existing customers who deployed Flash or Flex applications. IBM Tealeaf does not support new customers who use these technologies.

Note: Currently, IBM Tealeaf does not support UI capture and replay of multiple Flex or Flash applications on the same page. This limitation does not include multimedia content applications that do not require IBM Tealeaf UI capture or replay. If your application requires multiple Flex or Flash RIAs on the same page, each of which must be captured for replay, contact IBM Professional Services.

Note: IBM Tealeaf supports applications that use text-based data formats for communicating with the server. If your application uses binary data formats such as AMF, contact IBM Professional Services.

Supported protocols

IBM Tealeaf supports HTTP and HTTPS for request and response communication between client and server.

© Copyright IBM Corp. 1999, 2014

Verify that all Ajax send and receive communications are managed by HTTP or HTTPS.

The protocol of the client events is the same as the request that sent the page to the browser. If the client events must be sent by HTTPS, verify that the source page is delivered by HTTPS.

Captured form field values are transmitted in the same protocol in which the form page is transmitted to the server. If your form page is transmitted by HTTPS, the form field values are encrypted, too.

Note: IBM Tealeaf does not currently support any other protocols, such as HTTP streaming, HTTP push, or non-HTTP protocols such as RTMP. Verify that your application does not use non-HTTP/HTTPS protocols.

Before you begin development

Before you begin your IBM Tealeaf CX UI Capture j2 development initiatives, verify that the following information is available or was captured.

IBM Tealeaf access

If possible, deploy the web application on a server that is accessible to IBM Tealeaf developers. To effectively debug issues, IBM Tealeaf developers must have access to the application.

Third-party content

The following guidelines apply to third-party content.

Third-party domains

IBM Tealeaf cannot capture content that is managed by third-party domains. In a typical deployment, most of this content is not integral to application performance and can be re-requested at replay time without impacting overall application replay. If capture of dynamic versions of this content is required, contact IBM Professional Services.

Third-party plug-ins

Other than the Flash player, IBM Tealeaf does not currently support the replay of third-party plug-ins in Rich Internet Applications, such as Silverlight or Java™ applets.

Personnel

Personnel in the following roles need to be available to IBM Tealeaf for the CX UI Capture j2 deployment.

Role **Description**

Application architect
During deployment, IBM Tealeaf can make requests to the web application architect to identify locations in the application for calls to the IBM Tealeaf API. This individual is also useful for answering technical questions.

Business analyst
From the business perspective, it is important to know which areas of the application require monitoring and to what degree. This knowledge

directly translates into which UI elements, events, clicks, and other aspects of the application must be captured by IBM Tealeaf. During development, this individual might be able to identify qualified use cases for IBM Tealeaf CX UI Capture j2.

External resources

If application development is outsourced to a third-party, provide access to this team. IBM Tealeaf can work within any constraints for contracting external development teams.

Development considerations

During the development process, keep in mind the following considerations.

Development cycle

IBM Tealeaf assumes that the following basic production stages are in place.

- Development
- Testing
- Production

For more information about development goals during each of the stages, see Chapter 2, "CX UI Capture j2 installation and implementation," on page 9.

When to deploy CX UI Capture j2 during development

IBM Tealeaf recommends deploying IBM Tealeaf CX UI Capture j2 as early as possible in your development process.

In addition to curtailing the number of issues in a Production environment, CX UI Capture j2 can be useful for debugging issues with the application development process itself. Your web application and CX UI Capture j2 can work hand-in-hand to deliver a better overall result.

For example, changes to the client-side application, such as changes to the element identifiers, additions or modifications of UI widgets can impact capture and replay of the client-side UI. Identifying these changes via IBM Tealeaf CX UI Capture j2 in a Development environment facilitates faster resolution and a better replay experience once the application is deployed in the Production environment.

You can also use IBM Tealeaf CX UI Capture j2 privacy rules configuration to identify and protect sensitive client data before deployment to the live site.

By including IBM Tealeaf as part of the main development effort for your web application, CX UI Capture j2 can be integrated into the change management and scheduled maintenance processes for your enterprise.

Development and Test environments

Where possible, configure all Development or Testing environments to mirror the Production environment. Apply the same web server settings and permissions that are configured in the Production server to the Development environment or Test environments.

Note: Never deploy IBM Tealeaf CX UI Capture j2 directly into a Production environment.

Performance management

Integrate IBM Tealeaf CX UI Capture j2 into the Performance Management test suite for your enterprise.

Performance measurement of IBM Tealeaf CX UI Capture j2 JavaScript can help in configuring the size and frequency of the UI Capture posts.

Application content

When developing your web application, keep in mind the following considerations.

Unique identifiers

All user interface elements of the web application must have unique identifiers. Where possible, implement the application with unique identifiers for each UI element.

If unique identifiers are not available, IBM Tealeaf relies on XPath identifiers. However, this solution might not be 100% accurate, and use of XPaths can increase the size of each CX UI Capture j2 message and the total cost of storing the data.

Note: IBM Tealeaf CX UI Capture j2 does not automatically check for uniqueness of identifiers. Non-unique identifiers must be blacklisted manually through the configuration.

Dynamically transmitted GUIDs

If the application generates client-side dynamic GUIDs that are transmitted over the network, replay of captured sessions can be affected. Timestamps and changes in application behavior due to user agent, visitor locale, and other dynamic factors can have effects on replay.

Depending on the variations, effective replay might require IBM Professional Services to develop complex rules to match the dynamically generated POST data, remap server hosts, and more.

Note: Wherever possible, avoid client-side dynamically generated identifiers. While dynamic identifiers can be tracked by IBM Tealeaf, managing them is another layer of complexity in configuration.

Do not use Document Object Model keywords as form field names

This requirement is best illustrated by example.

The following is a specification of an HTML form that contains three input fields, the name of which uses the HTML and Document Object Model keywords `action`, `type`, and `tagName`.

```
<form id="my_form" method="get" action="submit.php">
  <input type="text" name="action" value="hotel" />
  <input type="text" name="type" />
  <input type="text" name="tagName" />
</form>
```

When CX UI Capture j2 attempts to report the standard properties (such as `type`, `action`, or `tagName`) of the form node on which a user event occurred, the value is

typically a defined string or an undefined or null value. In this case, the target is a form element, whose input field names are exported by the browser as properties of the form object. As a result, the form's `action`, `type`, and `tagName` properties refers to the respective input field, which is an object. Any JavaScript that attempts to access one of these native properties retrieves the input field object instead.

Note: Do not use any reserved Document Object Model keyword as the name of a form field.

Application objects

The following sections describe aspects of the application you are developing and how they might affect IBM Tealeaf CX UI Capture j2.

In general, do not cancel JavaScript events in your application. Allow all events of child elements to bubble up to the document's root element.

Ajax

- Where possible, do not use anchor tags as the value on which to change Ajax driven views, modals, or fly-out windows.
- In Ajax calls, do not make the request parameters or response data difficult to review and analyze. Create a request body that clearly shows what is being requested and where it is placed in the root document. The same applies to the response.
- For Ajax calls that result in updates to the page in the form of new user input elements such as drop-down lists or text boxes, IBM Tealeaf needs to be notified so that user interactions with these newly inserted elements can be captured. The CX UI Capture j2 library provides a simple API for this purpose.

Custom UI controls

Note: Capture and replay of custom UI controls might require a custom solution. For more information, contact IBM Professional Services.

HTTP cache control headers

Depending on your visitors' browser settings, CX UI Capture j2 JavaScript might be stored in the local cache for a lengthy period. If you deploy an updated version of CX UI Capture j2, your visitors' browsers might still pull the CX UI Capture j2 JavaScript from the local cache, which can cause significant problems if configuration changes were required for UI Capture to work with your application.

To manage this potential problem, use HTTP cache control headers for your web server and configuring the use of conditional HTTP requests through the standard `Last-Modified` HTTP header. Any updates to the CX UI Capture j2 files on the server are automatically propagated by the HTTP caching mechanism.

Note: HTTP caching requires more configuration and can cause issues if not configured properly. Consult with your IT staff before you enable these changes.

Note: If you cannot make the configuration changes to your web server, you might be able to manage updates by placing a date stamp in the file name of the IBM Tealeaf CX UI Capture j2 files, such as `tealeaf_110310.js`. However, these file name changes must be coordinated with your web development team and do not always trigger correct caching behavior.

References:

- http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html#sec13
- http://httpd.apache.org/docs/2.1/caching.html

Private data

IBM Tealeaf provides many options for managing data privacy throughout the capture, replay, and reporting functions.

Data can be blocked or masked through CX UI Capture j2, at the point of capture in the CX Passive Capture Application, or by using the Privacy session agent in the pipeline on the Processing Server.

Identify the data that requires privacy early in the development process and test your privacy configuration in the Development and Testing environments. Where possible, seek to minimize the volume of data that requires privacy management.

Note: When CX UI Capture j2 data is blocked or masked by using IBM Tealeaf privacy, replay can break if the web application is not designed to accept the altered content.

- For more information about how IBM Tealeaf secures data, see "Managing Data Privacy in Tealeaf CX" in the *IBM Tealeaf CX Installation Manual*.

Frequency of posts

IBM Tealeaf enables the configuration of the size and frequency of POSTs to the IBM Tealeaf target page.

- During Development, you can send small and frequent POSTs so that you can monitor activities at a granular level.
- In a Production environment, minimize the number of POSTs, each of which has a data overhead.

  Note: Ideally, the trigger for sending each POST in a Production environment is the maximum size of the post.

The size and frequency of these posts can be configured by parameter to control:

- The maximum allowed number of events in the buffer. If this maximum is exceeded, the queued events are posted.
- A timer value for periodic flushing of the event queue, regardless of the buffer size

UI events overridden by CX UI Capture j2

In some implementations, IBM Tealeaf can be configured to override specific user interface events.

Verify that there is no conflict between the IBM Tealeaf overrides and events that are used or monitored by your application.

See Chapter 2, “CX UI Capture j2 installation and implementation,” on page 9.

CX UI Capture j2 deployment

Placement of CX UI Capture j2

For best results, place the CX UI Capture j2 configuration file at the beginning of the page in the HTML head section.

Placement in other sections of the page is also supported.

See Chapter 2, “CX UI Capture j2 installation and implementation,” on page 9.

JavaScript

The following guidelines apply to working with JavaScript.

Modifications

Avoid changing any IBM Tealeaf configuration settings unless you are sure of the reasons for the change and its effects. Some options can have significant effects on performance.

Commenting

Monitor any changes to the JavaScript files by adding detailed comments. Periodically, IBM Tealeaf provides updates to the JavaScript files, and integrating these changes with the changes for your company is easier with thorough commenting.

Special functions

Verify that special page-level functions are not overwritten. These functions include `onload`, `onunload`, and `onreadystatechange`. If overwriting is required, then always "pass on" the call. For example:

```
if( typeof document.onreadystatechange == "function" ) {
        document.originalReadyStateChange = document.onreadystatechange;
    }
    else {
        document.originalReadyStateChange = null;
    }

    document.onreadystatechange = function() {
        //    Do my work here

        //    Call the original ReadyStateChange
        if(document.originalReadyStateChange) {
            document.originalReadyStateChange();
        }
    };
```

Create custom events

By default, IBM Tealeaf CX UI Capture j2 monitors user interactions with the application by capturing the standard Document Object Model events. IBM Tealeaf CX UI Capture j2 can also be used to monitor specific aspects of the application such as client-side performance metrics, client-side input validation messages, and any other data relevant to the application.

To capture these aspects and other client-side activity, the application developer can use a simple IBM Tealeaf API to notify the library of any data that must be included in the IBM Tealeaf POST.

Custom events are best implemented in a centralized location where the application processes its input validation or performance metrics, for example.

Note: Creating custom events to insert data in submissions to IBM Tealeaf increases the volume of data that is stored for each session. Create custom events only for business-critical data. Sending large volumes of UI data increases network traffic and storage requirements within IBM Tealeaf and might affect application performance.

See Chapter 6, "CX UI Capture j2 Public API Reference," on page 57.

Next steps

For extra information about specific practices to apply to your web application development and IBM Tealeaf CX UI Capture j2 implementation, contact IBM Professional Services.

Chapter 5. CX UI Capture j2 reference

The IBM Tealeaf CX UI Capture j2 library configuration is an anonymous JSON object container. The library configuration includes different sections.

- Core configuration object: Contains configuration information about Document Object Model events to which the library listens. See "Core configuration."
- Services configuration object: Contains configuration information about the individual services that extend the core and provide the necessary functionality that is required by the library. See "Services configuration" on page 46.
- Modules configuration object: Contains configuration information about the modules that are enabled in the library. See "Modules configuration" on page 54.

You can also configure keys for capturing localStorage data.

Core configuration

The core configuration object contains basic information about the DOM events to which the IBM Tealeaf CX UI Capture j2 library listens.

```
TLT.init({
   /**
    * Base configuration for the core.
    * It specifies which modules should listen to which events and the
    * moduleBase to dynamically load additional modules
    * @type {Object}
    */
   core: {
       /**
        * The ieExcludedLinks entry specifies a set of links that should
        * not trigger the beforeunload event, which destroys the UI
        * Capture library. This addresses a known issue with Internet
        * Explorer
        */
       ieExcludedLinks: ["a.ignore"],

       /**
        * This is the module configuration section contains options
        * relevant to the core.
        * It contains the events to which the modules attempt to listen.
        * NOTE: Please do not use this section for module-specific
        * configuration, which you must access inside of your modules.
        * @type {Object}
        */
       modules: {
           performance: {
               enabled: true,
               events: [
                   {
                       name: "load",
                       target: window
                   },
                   {
                       name: "unload",
                       target: window
                   }
               ]
           },
           replay: {
               enabled: true,
```

© Copyright IBM Corp. 1999, 2014

```
                events: [
                    /* Lifecycle events - not optional */
                    {
                        name: "load",
                        target: window
                    },
                    {
                        name: "unload",
                        target: window
                    },
                    /* User interaction events - not optional */
                    {
                        name: "change",
                        target: changeTarget,
                        recurseFrames: true
                    },
                    {
                        name: "click",
                        recurseFrames: true
                    },

                    /* Dwell time and previous state */
                    {
                        name: "focus",
                        target: "input, select, textarea, [contenteditable]",
                        recurseFrames: true
                    },
                    {
                        name: "blur",
                        target: "input, select, textarea, [contenteditable]",
                        recurseFrames: true
                    },

                    /* Hash change - optional */
                    {
                      name: "hashchange",
                      target: window
                    },

                    /* ClientState events - not optional */
                    {
                        name: "resize",
                        target: window
                    },
                    {
                        name: "scroll",
                        target: window
                    },

                    /* Mobile DOM events - optional */
                    {
                        name: "orientationchange",
                        target: window
                    },
                    {
                        name: "touchend"
                    }
                ]
            }
        }
    },
},
/* Following settings are related to the TLT.getSessionData() Public API */

/**
* Set the sessionDataEnabled flag to true only if it's OK to expose Tealeaf
* session data to 3rd party scripts.
```

```
*/
sessionDataEnabled: false,

/**
* The sessionData object specifies the query parameter name OR cookie name to
* get the session id value from. An actual deployment would only specify one
* of the methods to obtain the session id. If both are specified, the
* sessionQueryName takes precedence over sessionCookieName. If neither is
* specified, an internal default of "TLTSID" will be looked up in the cookies.
*/
sessionData: {
    /**
    * Specify sessionQueryName only if the session id is derived from a
    * query parameter.
    */
    sessionQueryName: "sessionID",

    /* Optionally, specify the query string delimiter. Default is & */
    sessionQueryDelim: ";",

    /* The cookie being used for sessionization in Tealeaf. */
    sessionCookieName: "jsessionid",

    /**
    * Optionally, specify if the value needs to be hashed to derive the
    * session ID.
    */
    sessionValueNeedsHashing: true
}
   /**
    * The framesBlacklist array specifies the CSS selectors which correspond to the
    * frame/iframe elements that should not be accessed.
    */
framesBlacklist: [
    ".tlBlock"
]
},

   /* End of core configuration */

   // Services configuration

   ...

   // Module specific configuration

   ...

});
```

Core

In **Core**, you define properties to assure capture of UI events.

Note: Do not modify these settings unless directed to do so by IBM Tealeaf. For more information, contact IBM Professional Services.

Internet Explorer links

In Internet Explorer, clicking an `href=javascript` link can trigger the `beforeunload` event. If this link does not trigger a navigation change, then the CX UI Capture j2 library is destroyed prematurely, and subsequent UI events on the page are not captured.

To address this issue, you can specify a series of CSS Selectors that resolve to HTML elements, which, when clicked, do not trigger the `beforeunload` event.

In the `ieExcludedLinks` field, you can specify these HTML elements as an array of CSS selectors. In the example below, an `<a>` anchor with a specified class (ignore) and an anchor with a specific class (foo) containing a span element are specified. If the user clicks either of these element types, then the resulting `beforeunload` event in Internet Explorer is ignored.

```
ieExcludedLinks: ["a.ignore", "a.foo span"],
```

Configure client events by module

The modules subsection of the core configuration specifies the events to which each module listens.

Note: Do not change the default settings.

An example configuration for an individual user interface event is the following configuration.

```
{
    name: "load",
    target: window
},
```

Where:

- `name` identifies the W3C name of the user interface event.
- `target` identifies the client object that is the target of the event.
  - The target can be a Document Object Model element such as document or window.
  - The target can also be a string that contains comma-separated CSS selectors to be used as the targets.
  - If target is not specified, it defaults to the document element.

    Note: To disable an event, comment out or remove the entry. Some events are not optional and must remain enabled for correct operation of the library.

For some user interface events, configuration might be in the following format:

```
{
    name: "click",
    recurseFrames: true
},
```

- If `recurseFrames` is set to `true`, then evaluation is applied to all frames and subframes on the page that originate from the same domain as the page.

Performance

The performance module parses the W3C Navigation Timing performance object and includes this information as the performance message in the JSON data stream that is sent back to the IBM Tealeaf capture server.

Note: For correct operation, do not modify this configuration. If the performance module is not required, remove this section from the core configuration.

Replay

The replay module provides the underlying data in form of JSON messages that enable features such as:

- Rich Internet Application (RIA) Replay: Browser Based Replay and CX RealiTea Viewer.
- IBM Tealeaf cxOverstat usability functionality; heatmaps, form analytics, and attention maps.

  See "cxOverstat User Manual" in the *IBM Tealeaf cxOverstat User Manual*.
- Step-based eventing.

  See "Step-Based Eventing" in the *IBM Tealeaf Event Manager Manual*.

The following events are tracked for replay purposes on the client. Most of these events are required for replay.

Note: Support for all of these replay events is provided in Browser-Based Replay only. See "CX Browser Based Replay" in the *IBM Tealeaf cxImpact User Manual*.

A subset of replay events might work in the CX RealiTea Viewer. See "RealiTea Viewer - Replay View" in the *IBM Tealeaf CX RealiTea Viewer User Manual*.

Property
: **Description**

`"load"` Page load event.

Note: This event is required.

`"unload"`
: Page unload event.

Note: This event is required.

`"change"`
: Control change event.

Note: This event is required. By default, it is configured to track all HTML elements that support the change event.

- Optionally, you can set `recurseFrames` to `true` to scan all frames and subframes of the page for these events.

`"click"`
: Click events on the page.

Note: This event is required.

- Optionally, you can set `recurseFrames` to `true` to scan all frames and subframes of the page for these events.

`"focus"`
: Focus events on the control.

Note: This event is required.

- Optionally, you can set `recurseFrames` to `true` to scan all frames and subframes of the page for these events.

`"blur"` Blur events on the control.

Note: This event is required.

- Optionally, you can set `recurseFrames` to `true` to scan all frames and subframes of the page for these events.

"hashchange"
When enabled, this option generates screen view events when a hash change was identified in the URL of the page.

- A `screenview` is defined as a change in state for the overall page. For example, if a page has multiple tabs, each tab can be defined as a separate screen view for the page.
- When this option is enabled, a screen view event is inserted in the session data, and UI events that are captured by UI Capture can be organized beneath the screen view on which they occurred.
- See Tealeaf JSON Object Schema Reference.
- If your web application does not use screen views, disable this option.

"resize"
Resize events on the client.

See Tealeaf JSON Object Schema Reference.

"scroll"
Scroll events on the client.

See Tealeaf JSON Object Schema Reference.

Note: Depending on your application, tracking windows that scroll can generate a significant number of events.

Note: Replay of scroll events is supported for mobile sessions in Browser Based Replay only. See Search and Replay for Mobile Web.

"orientationchange"
Orientation change events for mobile devices.

See Tealeaf JSON Object Schema Reference.

Note: These events are replayed only if you licensed IBM Tealeaf CX Mobile. For more information, contact your IBM Tealeaf representative.

"touchend"
Touch end events for mobile devices.

See Tealeaf JSON Object Schema Reference.

Note: These events are replayed only if you licensed IBM Tealeaf CX Mobile. For more information, contact your IBM Tealeaf representative.

Note: Pinch and zoom gestures can be captured on the iOS platform only. They cannot be captured on the Android platform.

Services configuration

The Services configuration object contains options that are used by the individual services that are part of the IBM Tealeaf CX UI Capture j2 library.

Note: When integrating the library with a new application, you typically modify only few of these configuration items.

```
/**
 * The service configuration for options used inside of the services.
 * @type {Object}
 * @required
 */
services: {
  /**
   * The queue configuration is an object containing an array
   * of queues. It must specify at least one default queue.
   * @type {Array}
   */
  queue: {
    /**
     * Flag to enable asynchronous XHR on page unload.
     * WARNING: Enabling this may result in incomplete or missing data.
     * @type {Boolean}
     * @default false
     */
    asyncReqOnUnload: false,

    /**
     * Array of queues. Must contain at least one default queue.
     * @type {Array}
     */
    queues: [
      /**
       * A default queue configuration.
       */
      {
        /**
         * The queue ID for the default queue is "DEFAULT".
         * For other queues, it could be any string to identify the
         * queue.
         * @type {String}
         */
        qid: "DEFAULT",
        /**
         * The endpoint on the server where the messages should be
         * sent. This value should correspond to a Tealeaf Target page
         * identifier.
         * @type {String}
         */
        endpoint: "TealeafTarget.php",
        /**
         * For maxEvents, you could specify the number of queued events
         * after which the queue should flush and send all of them to
         * the server.
         * @type {Number}
         * @default 25
         */
        maxEvents: 25,
        /**
         * The interval at which the queue should be flushed
         * automatically.
         * (in milliseconds)
         * If this option is not present or is set to 0, the timer is
         * disabled.
         * @type {Number}
         * @optional
         * @default 0
         */
        timerInterval: 0
        /**
         * Flag indicating if a cross domain POST should be used. If
         * enabled then the crossDomainFrameSelector should specify
         * the iframe or frame element on the page that has been
         * configured to POST requests.
```

```
            * @type {Boolean}
            * @optional
            * @default false
            */
           crossDomainEnabled: false,
           /**
            * A CSS selector that resolves to a single frame or iframe
            * element on the page that has been configured to POST
            * requests.
            * @type {string}
            * @optional
            */
           crossDomainFrameSelector: "#uicXdomain"
         }
    ]
  },
```

Queue service configuration

Through the Queue service configuration, you can add extra event queues for processing and configure event queue properties that are based on traffic volume.

Note: Adding an extra queue is an advanced option and is not required in most deployments.

Property
: **Description**

asyncReqOnUnload
: Enables asynchronous XHR on page unload.

  Note: Enabling this option may result in incomplete or missing data.

qid
: Identifies the name of available event queues.

  Note: By default, one queue is defined as `DEFAULT`. Do not remove this queue or change its name.

endpoint
: Identifies the target to which messages are sent. This value corresponds to an IBM Tealeaf target page identifier. Sample targets for JSP and PHP environments are provided with the UI Capture distribution.

maxEvents
: The maximum number of events in a queue. If this number of events is exceeded, the queue is automatically flushed, and all queued events are sent to the target for capture by IBM Tealeaf.

  Note: Adjust the maximum number of events to a reasonable value, typically between `20` to `50`. The maximum allowed value is `100`.

timerInterval
: The interval in milliseconds at which event queues are flushed automatically.

  If this option is not present or is set to 0, the timer is disabled.

  Note: Keep this setting disabled, unless a more immediate flushing of data is required, such as when IBM Tealeaf users are shadow-browsing a visitor's session.

Browser service configuration

Through the Browser service configuration, you can specify browser objects and elements that have dynamic or duplicate HTML IDs.

```
browser: {
    /**
     * sizzleURL is an URL to the sizzle.js file that contains the
     * Sizzle library. It is used in some older browsers to get
     * elements via a CSS selector.
     *
     * In such cases, the w3c browserService will conditionally load
     * this file.
     *
     * This setting is only valid when the W3C flavor of the
     * browser service is selected.
     *
     * @type {String}
     * @optional (only needed for w3c browserService and if jQuery or
     *           Sizzle is not defined on the window)
     */
    sizzleURL: "/scripts/legacyIESupport/sizzle.js",
    /**
     * sizzleObject is the location of the Sizzle object on the page. If
     * this setting is not specified, it is assumed that the Sizzle object
     * will be in the global scope. i.e. window.Sizzle
     *
     * This setting is only valid when the W3C flavor of the browser
     * service is selected.
     *
     * @type {String}
     * @optional (only needed for W3C browserService and if jQuery
     *            is not defined)
     */
    sizzleObject: "My.location.Sizzle",

    /**
     * jQueryObject is the location of the jQuery object on the page. If
     * this setting is not specified, it is assumed that the jQuery object
     * will be in the global scope. i.e. window.jQuery
     *
     * @type {String}
     * @optional (only needed if jQuery is not defined on the window)
     */
    jQueryObject: "My.location.jQuery",
    /**
     * To specify elements whose HTML ids cannot be used as identifiers
     * or in xPath generation. Could be a string object with one or two
     * properties, "regex" (mandatory) and "flags". These are specified
     * as strings. The regex that is evaluated against the node id.
     * @type {Array}
     * @optional
     */
    blacklist: [ "duplicateid", {regex: "/password|pass|pin|tan/",
    flags: "gi"} ],
    /**
     * If the application uses custom attributes as an ID replacement,
     * you may specify the attribute name here.
     * @type {Array}
     * @optional
     */
    customid: [ "mycustomid" ]
},
```

The Sizzle JS engine is required for correct operation of the W3C flavor of the library in legacy browsers (e.g. IE 7) that do not have native support for CSS selector based querys. Sizzle is included in jQuery by default so if the application has a newer version of jQuery, the separate include of Sizzle

```
may not required.

The library has multiple ways to specify the location of the Sizzle engine.
In order of preference these are:

sizzleURL - If specified, this will be used and subsequent mechanisms for
locating Sizzle will be ignored.
sizzleObject - If specified, this will be used instead of window.Sizzle
window.Sizzle
jQueryObject - If specified, this will be used instead of window.jQuery
window.jQuery
```

Property

Description

sizzleURL

You can use this configuration item to specify the URL to the `sizzle.js` file that contains the Sizzle library. This library is used by IBM Tealeaf CX UI Capture j2 to evaluate CSS selectors on older browsers.

Note: This configuration item is needed only for the W3C browser service and if JQuery or Sizzle is not defined on the window object of your web application.

- The Sizzle library is required for some older browsers to acquire document elements that use a CSS selector. If the client browser is one of the affected versions, the W3C browser service can automatically load this file from your web server.
- The list of object can be specified as comma-separated values, which can be fixed strings or regular expressions. These values are evaluated against the node ID.

blacklist

You can optionally specify a set of document elements whose HTML IDs are not guaranteed to be static or unique. These HTML IDs of these blacklisted objects are not used to identify the elements and alternate mechanisms, such as a `customid` (if configured) or an xpath.

customid

If your web application uses a custom attribute that can be used to uniquely identify the elements, you can specify the attribute name here.

Message service configuration

You use the Message service configuration to define options such as privacy rules applied to the JSON messages submitted to the IBM Tealeaf server for capture.

```
message: {
        /**
         * Define privacy rules to modify reported values in the messages
          * sent to the server.
         * @type {Array}
         * @optional
         */
        privacy: [
            /**
             * An example privacy rule.
             */
            {
                /**
                 * Specify the targets to which this rule should apply.
                 * @type {Array}
                 */
                targets: [
```

```
                    {
                        id: "htmlid",
                        idType: -1
                    },
                    {
                        id: "myid=custom",
                        idType: -3
                    }
                ],
                /**
                 * The maskType defines how the value should get
                 * transformed.
                 * maskType 1: The value gets set to an empty string
                 * maskType 2: The value gets replaced with the fixed
                 *             string "XXXXX".
                 * maskType 3: The value gets replaced by a mask where:
                 *     each lowercase character gets replaced by "x",
                 *     each uppercase character by "X",
                 *     each number by "9"
                 *     each symbol by "@"
                 *     e.g. "HelloWorld123" becomes "XxxxxXxxxx999"
                 * maskType 4: The value gets replaced by the return
                 *             value of
                 *  a custom function that must be specified as
                 *  maskFunction.
                 * @type {Number}
                 */
                maskType: 3
            },
            {
                targets: [
                    {
                        id: "[[\"HTML\",0],[\"BODY\",0],[\"SELECT\",0]]",
                        idType: -2
                    }
                ],
                maskType: 4,
                /**
                 * A custom mask function that replaces the reported value
                 * by its return value ("masked", in this example)
                 * @param  {String} value The value to replace.
                 * @return {String}         Could return any string.
                 */
                maskFunction: function (value) {
                    return "masked";
                }
            }
        ]
    }
},
```

Privacy configuration for CX UI Capture j2

IBM Tealeaf CX UI Capture j2 enables the blocking and masking of sensitive information within the client browser, before the data is forwarded to IBM Tealeaf for capture, while it allows the data to be forwarded to your web servers for normal processing. Sensitive data that was cleansed through CX UI Capture j2 never reaches IBM Tealeaf, which ensures that your customer's interactions are secure in CX UI Capture j2.

- CX UI Capture j2 enables the blocking of user input data by element ID, name, or xpath.
- Masks can be expressed as explicit strings, replacements for character types, or custom functions.

- For more information about how data privacy is managed throughout the IBM Tealeaf system, see "Managing Data Privacy in Tealeaf CX" in the *IBM Tealeaf CX Installation Manual*.

  Note: If you have questions about implementing data privacy in CX UI Capture j2, contact IBM Professional Services.

To specify a privacy rule, you must define:

- The type of identifier.
- The targets to which the rule applies.
- The type of masking to apply to the targets.

Specifying a privacy rule

In the configuration, a single privacy rule is specified within the `privacy` object by using the following configuration template.

```
{
    targets: [
        {
            id: "htmlid",
            idType: -1
        }
    ],
    maskType: 3
}
```

Specifying targets

To specify a target, you must specify the following properties.

Note: You can specify multiple `id` or `idType` targets for each masking rule.

Property

Description

`id` The identifier for the target element. This value is specified according to the `idType` value.
In the configuration file, you can use a regular expression to specify matching identifiers. For example, the following target configuration matches all HTML identifiers that end with `_pii`:

```
message: {
    privacy: [
        {
            targets: [
                {
                    id: { regex: ".+_pii$" },
                    idType: -1
                },
            ],
            "maskType": 3
        }
    ]
}
```

`idType` The type of identifier. The following types are supported:

Note: Values for `idType` are recorded as negative numbers.

- `-1` - HTML ID
- `-2` - xpath identifier

- `-3` - HTML name or other element attribute identifier

CSS selector

In the configuration file, you can also specify CSS selector values to match CSS elements for privacy masking. In the example below, the designated privacy rule is applied to all password input fields:

```
message: {
    privacy: [
        {
            targets: [
                "input[type=password]"
            ],
            "maskType": 3
        }
    ]
}
```

Specifying mask type

IBM Tealeaf CX UI Capture j2 supports the following mask types (`maskType` values):

Table 2. Privacy configuration for UI Capture j2

Value	Description	Example	Masked Example
1	Value is blocked and replaced by an empty string.	`"HelloWorld123"`	`""`
2	Value is masked with a fixed string of `X`'s	`"HelloWorld123"`	`XXXXX`
3	Value is masked according to the following parameters: • a lowercase letter is replaced by `x`. • an uppercase letter is replaced by `X`. • a numeral is replaced by `9`. • a non-alphanumeric value is replaced by `@`.	`"HelloWorld123"`	`"XxxxxXxxxx999"`
4	Custom function **Note:** A masking function must be defined as `maskFunction`.	`"HelloWorld123`	Depends on the defined function

Serializer service configuration

Through the Serializer service configuration, you can specify which JSON parser and serializer functionality to use.

The library uses JSON as its message format. All modern browsers have built-in support for serializing and parsing JSON. However, legacy browsers (for example, Internet Explorer 6 and 7) do not have this built-in support. For such browsers, a third-party library that provides JSON serialization and parsing capabilities is required. By default, the CX UI Capture j2 library relies on the built-in JSON capabilities of the browser where available, and defaults to a simplified internal

implementation in other cases. If your application or IT environment mandates use of specific JSON parser or serializer, you can specify it using this configuration.

Note: JSON serialization and parsing are fundamental to the library's ability to function and produce correct output. Thorough cross-browser testing is necessary if any changes are made to this configuration.

```
serializer: {
        json: {
                /**
                 * If this flag is enabled, the simplified internal
                * implementation of the JSON serializer and parser will
                 * be used if none of the specified parsers and
                 * stringifiers are available.
                 *
                 * If this flag is disabled, the library will shutdown
                 * if the specified JSON serializer and parser are
                 * not available.
                 */
                defaultToBuiltin: true,

                /**
                 * This array specifies an ordered sequence of JSON
                 * parsers that should be used to parse a serialized
                 * JSON object.
                 *
                 * The parser must support the following interface:
                 *     var jsObject = parse(jsonString);
                 */
                parsers: [ "JSON.parse" ],

                /**
                 * This array specifies an ordered sequence of JSON
                 * serializers that should be used to serialize a
                 * JavaScript object.
                 *
                 * The serializer must support the following interface:
                 *     var jsonString = stringify(jsObject);
                 */
                stringifiers: [ "JSON.stringify" ]
        }
}
```

Modules configuration

The Modules configuration object contains options that are used by individual modules that are enabled in the IBM Tealeaf CX UI Capture j2 library. These options correspond to the available options of the W3C navigation timing specification.

See http://www.w3.org/TR/navigation-timing/#sec-navigation-timing-interface

```
modules: {
 performance: {
     /**
      * The calculateRenderTime setting only applies on browsers
      * that do not support the W3C Navigation Timing API.
      *
      * If enabled, the library will calculate and report the render
      * time as a difference between the library load timestamp and
      * the page load timestamp.
      *
      * To ensure an accurate calculation ensure the library is
      * initialized as early as possible in the page load cycle. Due
      * to these limitations the render time calculated in this
```

```
         * manner will always be less accurate than what is reported by
         * the browser.
         */
        calculateRenderTime: true,
        filter: {
            navigationStart: true,
            unloadEventStart: true,
            unloadEventEnd: true,
            redirectStart: true,
            redirectEnd: true,
            fetchStart: true,
            domainLookupStart: true,
            domainLookupEnd: true,
            connectStart: true,
            connectEnd: true,
            secureConnectionStart: true,
            requestStart: true,
            responseStart: true,
            responseEnd: true,
            domLoading: true,
            domInteractive: true,
            domContentLoadedEventStart: true,
            domContentLoadedEventEnd: true,
            domComplete: true,
            loadEventStart: true,
            loadEventEnd: true
        }
      }
   }
});
```

Performance

In the Performance section, you configure the client interface events that are tracked for performance metrics.

Filter

Using the filters available in the Performance section, you configure the Performance-related events in the client to monitor by CX UI Capture j2. The list of provided metrics that you can filter out is pre-defined by the W3C navigation timing specification.

For more information, visit http://www.w3c.org.

By default, performance data is not captured, as it can have a significant impact on bandwidth. As a result, the default configuration filters all performance data (performance metric: `true`).

Note: When an individual filter is enabled, performance data that is related to that property is not captured by CX UI Capture j2.

To enable tracking of any performance property, set its value to `false`.

localStorage configuration

Many browsers support localStorage, which is faster and results in larger amount of data stored locally on the user browser. To capture data, you use the CX UI Capture j2 SDK with localStorage capture to define the keys to be captured.

The data is stored in key-value pairs, and a web page can access only data that is stored by itself. The localStorage object stores the data with no expiration date. The data is not deleted when the browser is closed, and is available the next day, week, or year.

The IBM Tealeaf CX UI Capture j2 localStorage captures only white list data.

Note: localStorage is not available in the general CX UI Capture j2 SDK. For localStorage capture, you need a CX UI Capture j2 SDK with localStorage capture.

To configure localStorage, you define a key to be captured. This key is added to the CX UI Capture j2 configuration. Nothing is captured if no keys are defined.

CX UI Capture j2 only captures keys if you use the `getItem() API` (for example, `localStorage.getItem("xyz")`). If you use dot notation nothing is captured (for example. `localStorage.xyz`).

Example:

```
modules: {
            performance: {
                filter: {

                }
            },
             replay: {
storageKeys: ["x", "y"]
             }
 }
```

IBM Tealeaf CX UI Capture j2 only captures the keys that you define and their values. CX UI Capture j2 captures localStorage items when they are accessed (for example, at `getItem()` call).

CX UI Capture j2 produces a type 8 event. It is then sent the JSON messages back to the server as is.

An example of type 8 capture follows.

For this example, the localStorage configuration is set to only capture the key `usrname`.

```
count: 9
fromWeb: true
offset: 10585
screenviewOffset: 10585
type: 8
webStorage:{
     key: "usrname"
     value: "ssingh"
}
```

Chapter 6. CX UI Capture j2 Public API Reference

The IBM Tealeaf CX UI Capture j2 library supports the public interfaces that are listed here.

In the development builds of the library, several APIs throw exceptions and log the exception to the console when the library is not initiated for API calls that depend on the library. In production builds of the library, the API call fails silently if the library was not initialized.

The following APIs throw this exception in development builds.

- `getSessionData`
- `logCustomEvent`
- `logExceptionEvent`
- `logScreenviewLoad`
- `logScreenviewUnload`
- `logScreenCapture`

TLT.init(object configObject)

This API initializes the IBM Tealeaf CX UI Capture j2 library with the specified configuration object. `init` execution occurs after the Document Object Model is loaded.

- For static deployments, the call to `init` can be included as part of the static JavaScript resource.
- For dynamic deployments, the call to `init` can be made from within the customer application.

Example

```
function callinit(){

  TLT.init();
}
```

TLT.rebind(/*optional*/ DOMElement root)

Rebind causes the IBM Tealeaf CX UI Capture j2 library to reprocess its event listeners. The application can use this API to notify the IBM Tealeaf CX UI Capture j2 library after the application made a dynamic update that added new user input elements to the Document Object Model.

The application can specify an optional root element from which the reprocessing occurs. If not specified, the reprocessing occurs from the document element. In a large and complex Document Object Model, specifying the exact subset that was updated with new user input elements improves efficiency.

© Copyright IBM Corp. 1999, 2014

Example

```
function callrebind() {

  TLT.rebind();

}
```

TLT.flushAll(void)

This API causes the buffered data that is gathered by the IBM Tealeaf CX UI Capture j2 library to be flushed to its configured endpoint URL.

Typically, flush is used in combination with the `setAutoFlush` API call to gain precise control over network usage behavior in highly tuned mobile apps.

Example

```
function callflushAll() {

  TLT.flushAll();
}
```

TLT.setAutoFlush(AutoFlushFlag flag)

This API is used to enable or disable the automatic flushing behavior of the library. Automatic flushing is enabled by default and is controlled by the configurable queue length.

To disable automatic flushing:

```
TLT.setAutoFlush(false)
```

To enable automatic flushing:

```
TLT.setAutoFlush(true)
```

Note: When automatic flushing is disabled, it is the application's responsibility to ensure the data that is buffered by the library is flushed out at appropriate intervals by starting the `flushAll()` API.

TLT.processDOMEvent(DOMEvent event)

This API is used to explicitly inform the IBM Tealeaf CX UI Capture j2 library of any Document Object Model event that the library was configured to listen to but is actively blocked by the application from bubbling.

In this case, the application or site can make an explicit API call. The argument is the DOMEvent object that is passed by the browser to the event handler.

It also applies to similar actions that prevent the CX UI Capture j2 library from automatically processing the event.

Example

```
$(document).ready(function(){
   $( ".disable" ).click(function( event ) {
     event.preventDefault();
     event.stopPropagation();

     // Notify Tealeaf
     if (TLT && TLT.processDOMEvent) {
```

```
        TLT.processDOMEvent(event);
    }

    $( "<div>" ).append( "default " + event.type + " prevented" ).
    appendTo( "#log" );
    });
  });
```

TLT.logCustomEvent(DOMString name, object customMsgObj)

The application can use this API to log a custom message in to the CX UI Capture j2 library's message stream.

The `customMsgObj` parameter to this function specifies the object to be serialized and passed as a custom message.

The custom message can be processed on the back-end for step-based eventing and reporting.

- See "Step-Based Eventing" in the *IBM Tealeaf CX Event Manager Manual*.
- See "Tealeaf Report Builder" in the *IBM Tealeaf cxImpact Reporting Guide*.

Example

```
function calllogCustomEvent(){

  TLT.logCustomEvent("logCustomEvent name",
  "This is to test the logCustomEvent API");
}
```

TLT.logExceptionEvent(DOMString msg, /*optional*/ DOMString url, /*optional*/ long line)

The application can use this API to log an error or exception message in to the IBM Tealeaf CX UI Capture j2 library's message stream.

Specifying the URL and line number of the error message is optional.

Example

```
function calllogExceptionEventh() {

  TLT.logExceptionEvent("This is to test the TLT.logExceptionEvent()","/
  APITesting/testRebind.html",101);
}
```

TLT.logScreenviewLoad(DOMString name, /*optional*/ DOMString referrerName, /*optional*/ DOMElement root)

The application can use this API to log a screen view load message in to the IBM Tealeaf CX UI Capture j2 library's message stream.

Specifying the referring screen view name and the root element for this view is optional.

Example

```
function calllogScreenviewLoad(){

  TLT.logScreenviewLoad("logScreenviewLoad name",
  "This is to test referrerName for logScreenviewLoad",
  document.getElementById("logScreenviewLoad"));
}
```

TLT.logScreenviewUnload(DOMString name)

The application can use this API to log a screen view unload message into the IBM Tealeaf CX UI Capture j2 library's message stream.

Example

```
function calllogScreenviewUnload(){

  TLT.logScreenviewUnload("logScreenviewUnload name");
}
```

TLT.getSessionData()

This API returns an IBM Tealeaf session data object. The IBM Tealeaf session ID information is included in this data.

An optional flag indicates if the session ID must be derived by hashing the session value in the object.

An example of a returned value follows.

```
{
    tltSCN: "PHPSESSID",
    tltSCV: "joese2pun5nus50p45j38hrak5",
    tltSCVNeedsHashing: true                // Optional
}
```

To enable this API, the library must be configured with the appropriate configuration settings. These settings inform the library from where the session ID information is to be derived.

If the API is not enabled in the configuration or the specified data cannot be read, then `null` is returned.

Example

```
function getSessionData() {

  var _sessionData = TLT.getSessionData();
  var sessionId = _sessionData.tltSCN;
  var sessionVal = _sessionData.tltSCV;

  verifySessionData(sessionId, sessionVal)

}
```

TLT.registerBridgeCallbacks

This API can be used to register callback functions, which are invoked by the CX UI Capture j2 library in specific instances.

Note: This is an advanced API, which can result in incorrect operation or loss of data if misused. This API should not be used in a hybrid application as the IBM Tealeaf native logging frameworks register and make use of the appropriate bridge callbacks.

Currently, three callback types are supported: `messageRedirect`, `screenCapture`, and `addRequestHeaders`.

Example to register a callback

```
TLT.registerBridgeCallbacks([
{enabled: true, cbType: 'screenCapture', cbFunction:
function (){tlBridge.screenCapture();}},
{enabled: true, cbType: 'messageRedirect', cbFunction:
function (data){tlBridge.addMessage(data);}}
])
```

Where `messageRedirectFunc` , `screenCaptureFunc`, and`addRequestHeadersFunc` are the callback functions.

messageRedirect

This callback can be registered to redirect and intercept messages from the UI Capture. When this callback is registered, the UI Capture library starts the callback function for each message before adding it to the queue. The callback function is passed two parameters: the serialized JSON message and the same message as a JavaScript object.

The callback function can consume the message entirely and return null or undefined. In this case, the UI Capture library discards the message.

The callback function can also modify the JavaScript object or return it unchanged. In this case, the UI Capture library queues the returned object in its internal queue.

This API must not be used in a hybrid application as the IBM Tealeaf native logging frameworks registers and makes use of this bridge callback.

Example of messageRedirectFunc

```
function messageRedirectFunc(msgStr, msg) {
    // Modify the mousedown type to a click
    if (msg.type === 4 && msg.event.type === "mousedown") {
        msg.event.type = "click";
    }
    return msg;
}
```

screenCapture

This callback function can be registered to enable a JavaScript API (see `logScreenCapture`) to allow a screen capture to be taken. The UI Capture library starts the callback function when the `logScreenCapture` API is started.

This API must not be used in a hybrid application as the IBM Tealeaf native logging frameworks registers and makes use of this bridge callback.

Example of screenCapture

```
function register_ScreenShot_Enable_CallBack() {
TLT.registerBridgeCallbacks([
    {
```

```
        enabled: true,
        cbType: "screenCapture",
        cbFunction: myCbFunction
    }
]);
}
```

addRequestHeaders

This callback function can be registered to enable a third party JavaScript to return custom HTTP headers that need to be set on the UI Capture libraries POST request.

Example of addReqHeadersFunc

```
count = 1;
function addReqHeadersFunc() {
    var headers = [
        {
            name: "X-Counter",
            value: count
        }
    ];
    if (count === 1) {
        // Add a recurring header the 1st time.
        headers.push({
            name: "X-Custom-Id,
            value: customId,
            recurring: true
        });
    }
    count++;
    return headers;
}
```

TLT.logScreenCapture

This API instructs the underlying native functionality to take a screen capture.

This functionality is only available when a `screenCapture` callback is registered with the UI Capture library by the native code.

Example

```
if (TLT && TLT.isInitialized()) {
    TLT.logScreenCapture();
}
```

Chapter 7. CX UI Capture j2 FAQ

This section includes frequently asked questions about how to acquire, implement, and use the IBM Tealeaf CX UI Capture j2 library that is provided by CX UI Capture j2.

Note: Except as noted, questions and answers in this section apply to IBM Tealeaf CX UI Capture j2 only.

Getting started

What is CX UI Capture j2?

CX UI Capture j2 is a JavaScript library, an example server-side application, and documentation. You use CX UI Capture j2 to capture the visitor's interactions with the web page (mouse clicks, keyboard input, scroll, and touch interactions, and so on), browser environment (window dimensions, and so on), and performance information (for example, render times).

CX UI Capture j2 collects this information and builds a message that is later posted to a target page on the application server. The target page does not perform any processing and returns an HTTP 200 status.

Posting can occur on `unload`, after a pre-defined time interval, or after the message queue reaches a pre-defined size.

Since the IBM Tealeaf core product captures all request and response interactions between the browser and the server, the posted CX UI Capture j2 data can be used to replay and report on the user interaction occurring within the page.

What versions of UI Capture are available?

UI Capture is available in two versions.

Beginning in Release 8.6, IBM Tealeaf introduced IBM Tealeaf CX UI Capture j2. CX UI Capture j2 submits data in an updated JSON schema. IBM Tealeaf CX UI Capture j2 is required for IBM Tealeaf cxOverstat.

Why do I need CX UI Capture j2?

CX UI Capture j2 captures user interactions with an application.

Some user experience data only exists in the browser and is never transmitted back to the server. This data includes render times, order of form field entry, form abandonment, and more.

Replay of Ajax web applications requires the inclusion of the CX UI Capture j2 JavaScript to capture the UI events generated as a result of the user interaction with the application. These captured UI events are used to stimulate the application during replay to match the Ajax requests/responses.

© Copyright IBM Corp. 1999, 2014

How do I implement CX UI Capture j2?

The following steps describe the basic process to implement the IBM Tealeaf CX UI Capture j2.

1. Determine which pages must be instrumented. Typically, these pages include JavaScript based functionality of interest. For more information, contact http://support.tealeaf.com.
2. Obtain the library from your software distribution. For more information on downloading IBM Tealeaf, see IBM Passport Advantage Online.
   - For the latest version information, see the Release Notes published on https://community.tealeaf.com/display/public.
   - You can access IBM Tealeaf Online Help through the Help menu in your IBM Tealeaf Portal. If you know your company login, use the URL above.
3. Review the documentation online or in the PDF supplied with the library. The online documentation is the most up-to-date.
4. Determine the capture options that are required for your web application. These options can vary for different pages on the website.
5. Based on the options you select, modify the JavaScript configuration files that are provided in the library.

   Note: IBM Tealeaf CX UI Capture j2 includes a GUI utility to help the configuration of your CX UI Capture j2 solution. This Configuration Wizard performs data validation and configures the IBM Tealeaf JavaScript depending on your selections. See Chapter 3, "CX UI Capture j2 Configuration Wizard," on page 23.
6. Modify the JavaServer page (`.jsp`) provided by IBM Tealeaf to meet your requirements. Or, you can choose to replicate it in a different server language, such as ASP or PHP.

   This page accepts the posts from the IBM Tealeaf JavaScript running on the browser and returns a success code. It does not evaluate or manipulate the received data, which is forwarded to and processed by your IBM Tealeaf system.
7. Modify the pages that call the CX UI Capture j2 library to include the IBM Tealeaf JavaScript.
8. In a test environment or on a non-production page on your production site, publish the IBM Tealeaf target page, the configured IBM Tealeaf JavaScript, and the modified pages.

   Note: This environment must be captured by IBM Tealeaf as a separate test environment instance or the production instance that is configured to keep the test traffic separate from your production traffic.
9. Test the web pages to ensure that the addition of the IBM Tealeaf JavaScript interacts correctly with the page.
10. Test the captured traffic on your IBM Tealeaf instance to verify that the UI traffic was captured.

    In some cases, you modify the replay rules to correctly replay these sessions. If you have any problems, review the documentation or contact http://support.tealeaf.com.
11. After you verify that all instrumented UI elements are being properly captured and processed by IBM Tealeaf, publish the changes to your production environment with your normal publishing and change control processes.

What is the size of CX UI Capture j2?

When the CX UI Capture j2 JavaScript is turned into a production version that is minified and compressed, it is approximately 22K.

Can I bundle the CX UI Capture j2 library with other JavaScripts?

No. Do not combine the CX UI Capture j2 library with other JavaScripts.

IBM Tealeaf recognizes that some customers bundle their JavaScript resources into a single file to simplify deployment of static content. However, due to the following risks and costs that are associated with bundling JavaScripts, IBM Tealeaf does not support bundling.

Risks

- The IBM Tealeaf library is mutually independent of any other code that runs in the context of your web application. Maintain this separation as a general policy.
- The browser treats each JavaScript include tag as a single unit of execution. When you bundle JavaScripts, a dependency is created between independent libraries. An error in one of the bundled scripts can prevent the other libraries from running. In the worst case scenario, an error in a peripheral script such as IBM Tealeaf can result in an application outage.
- If bundling is unavoidable, the customer is responsible for performing risk mitigation through exhaustive testing of the bundled package to verify that no problems are being introduced.

Other costs of bundling

- The bundled library needs to be tested concerning the application, as well as any required testing for individual libraries. The IBM Tealeaf JavaScripts can require configuration updates and multiple iterations to verify.
- Individual script changes or library upgrades require an entirely new bundle to be versioned, tested, and deployed.

Potential problems

Debugging problems in a bundled script is more complicated. Individual libraries cannot be easily enabled or disabled by using proxy tools such as Fiddler.

Note: If separating the IBM Tealeaf JavaScript from your bundled scripts is not an option, then in most situations the IBM Tealeaf JavaScript should be the last component of the bundle. Consult with IBM Professional Services to ensure effective management of the development and deployment of the bundle.

Will CX UI Capture j2 impact the performance or behavior of my website?

The library is designed to minimize the load on the web server and the web page running in the browser.

As part of the above implementation steps, the downloaded JavaScript is minimized and is cached by the browser.

Web server optimizations can further improve performance through server-side compression and caching settings.

See Chapter 2, “CX UI Capture j2 installation and implementation,” on page 9.

The server application accepting the posts is very simple and does nothing with the data.

The library posts add an extra hit or two to every page being captured by IBM Tealeaf. The size of the hit is small, normally increasing IBM Tealeaf storage requirements by less than 10%.

By design, the IBM Tealeaf system does not mix these hits with normal page hits in the page count reports.

With the library, your IBM Tealeaf solution may need to be configured with new replay rules to obtain replay fidelity.

Where can I get latest version of CX UI Capture j2?

To obtain latest version of the CX UI Capture j2 JavaScript code, go to IBM Passport Advantage Online.

What is being captured by CX UI Capture j2?

Depending on how you configure the library, CX UI Capture j2 can capture render times, browser dimensions, UI events (click, text input, scroll information, and so on) and more.

See Chapter 1, “IBM Tealeaf CX UI Capture j2,” on page 1.

I am having trouble with CX UI Capture j2 implementation. Where do I get support or help?

To contact Customer Support, visit the IBM Support Portal or connect through your Salesforce account.

Using CX UI Capture j2

How do I configure what is captured by CX UI Capture j2?

Currently, the IBM Tealeaf CX UI Capture j2 solution does not support dynamic or configurable capture levels.

CX UI Capture j2 monitors a predefined set of user interface events and properties, submitting messages for those that are detected and captured. See "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.

You can configure CX UI Capture j2 to capture custom events specific to your web application. See "UI Capture j2 Public API Reference".

What is captured from mobile web sessions?

Mobile web sessions are captured by using the same JSON schema that is used for desktop sessions, with some exceptions.

See "Tealeaf JSON Object Schema Reference" in the *IBM Tealeaf Client Framework Data Integration Guide*.

What if I find a bug with CX UI Capture j2?

If you find a bug, access the IBM Support Portal to report it.

Be prepared to provide the following information:

- CX UI Capture j2 version (found in `tealeaf.js` or `tealeaf.min.js` as `@version 2.0.0.664`)
- IBM Tealeaf Portal build number (at bottom of each Portal page)
- CX RealiTea Viewer version (if used)
- An IBM Tealeaf session not containing any sensitive information

How can I create events from CX UI Capture j2 data?

IBM Tealeaf CX UI Capture j2 submits data in JSON format. You can create events and step attributes from this data through the Browser-Based Replay interface, using step-based eventing. These event objects can be modified in the Event Manager as needed.

- See "Tealeaf JSON Properties" in the *IBM Tealeaf Client Framework Data Integration Guide*.
- See "Step-Based Eventing" in the *IBM Tealeaf CX Event Manager Manual*.
- See "Tealeaf Event Manager" in the *IBM Tealeaf CX Event Manager Manual*.

How can I search for CX UI Capture j2 data?

JSON-based data is indexed and available for search. Additionally, you can search for CX UI Capture j2 data if you created events and attributes to track and capture the data.

- See "Searching Session Data" in the *IBM Tealeaf cxImpact User Manual*.
- See "Step-Based Eventing" in the *IBM Tealeaf CX Event Manager Manual*.

How do I capture only `mouseover` events on menu?

IBM Tealeaf CX UI Capture j2 does not support capture of mouseover events.

How do I capture `keyup` events?

The JSON version of IBM Tealeaf CX UI Capture for AJAX does not capture `keyup` events from the client.

Instead, monitor the change events for the same objects.

How do I report on client-side validation/error messages?

IBM Tealeaf captures data that is exchanged between the browser and web server. To capture data that is not exchanged between the browser and web server, such as client-side validation, you create custom events.

As such, any validation or error condition that is detected on the server and communicated to the client is recorded in the IBM Tealeaf session.

Client-side validation is normally done by a JavaScript running on the user's browser. This validation does not result in a data exchange between the browser and web server. While client-side validation messages can be seen during replay of the session by virtue of replaying the same user input events that lead to the validation failures, they cannot be detected by IBM Tealeaf events that operate on the network traffic.

One solution is to make the client-side validation triggers part of the network traffic. Using an API provided with CX UI Capture j2, you can create a custom UI event in the browser by the same code that detects and triggers the validation message.

In the following example, a client-side validation function checks to verify that an input field is not empty.

```
function Validation_Check(user_input_element)
{
     if (!user_input_element.value) {
          // Input is empty, validation failed
          return FAILURE;
     }
     return SUCCESS;
}
```

To capture this validation trigger in an IBM Tealeaf session, you must create a custom event by using the `TLT.logCustomEvent()` method. The generated output can be captured from JSON format.

How do I force the sending of queued events?

By default, IBM Tealeaf queues captured UI events for transmission that is based on predefined thresholds. In some cases, it is necessary to force the sending of a set of queued events to the web server for capture by IBM Tealeaf.

For example, if a UI element such as a **Submit** button triggers the `onbeforeunload` event, the execution of that code can empty the queue of events before the `onclick` chain of events were properly submitted the queued events. In such a case, the `onclick` event that arrives after the completion of the site's event handler is lost.

To handle these situations, the library enables the forced flushing of the event queue during the unload phase.

To manually start this API in your code call `TLT.flushAll()`

For the first phase, the only desired metric is the Page Render time. Are there configuration settings that limit POSTs to just that event?

You can configure the UI Capture Configuration Wizard so that only performance data is submitted.

In the UI Capture Configuration Wizard, enable the Performance module only. Disable the Replay module. This configuration forces IBM Tealeaf CX UI Capture j2 to submit only performance data, which includes the Page Render time metric.

See "UI Capture j2 Configuration Wizard".

I use DHTML on my checkout form. Is CX UI Capture j2 going to capture the dynamically rendered DOM elements?

DHTML is managed by calling to `TLT.rebind()` to attach event handlers to your dynamically created input elements.

How do I get render times from CX UI Capture j2?

You can configure the UI Capture Configuration Wizard so that only performance data is submitted.

In the UI Capture Configuration Wizard, enable the Performance module only. Disable the Replay module. This configuration forces IBM Tealeaf CX UI Capture j2 to submit only performance data.

For all modern browsers that implement the W3C Navigation Timing specification, render times are reported through the Performance module.

See "UI Capture j2 Configuration Wizard".

Do I need to have IDs assigned to my DOM elements that I would like to capture?

If possible, assign unique IDs to the HTML elements that must be tracked. If doing so is not an option, assign unique IDs to the nearest ancestor HTML element.

In the absence of an ID, IBM Tealeaf CX UI Capture j2 builds an XPath like Document Object Model (DOM) path, which ends on the nearest ancestor with a valid HTML ID.

Can I customize CX UI Capture j2 JavaScripts?

No. IBM Tealeaf UI Capture provides configurable libraries for capturing user interface events for visitors to your j2 application, including many configuration options for tailoring the events to be captured and the methods by which they are captured.

These options are indicated in the product documentation and in the CX UI Capture j2 scripts.

IBM Tealeaf is disabled during Replay by setting Tealeaf and its objects to empty objects. This allow us to block Tealeaf from running during Replay. In the specific case where you added custom code, and did not disable it during Replay, while the rest of Tealeaf is disabled during replay through ReplaySplice JS, you may encounter JavaScript errors during replay, which causes replay to appear broken. If you add any custom code, you must make sure it is disabled in `ReplaySpliceJS` on the Replay Server.

Note: Modifications other than configuration changes to the JavaScript code of IBM Tealeaf CX UI Capture j2 that are not implemented by IBM Tealeaf according to an IBM Professional Services engagement voids the warranty concerning the software, and such modified code is not supported under the IBM Tealeaf Maintenance and Support program. For more information, contact IBM Professional Services.

How come I get cross-domain error messages in my browsers debug console related to iFrames on my site?

When the CX UI Capture j2 solution is initialized on any page of your web application, it tries to attach to any frame elements on the page. However, if these frame elements are not in the same domain as the web application, most browsers register this action as cross-domain access and forbid the execution of the CX UI Capture j2 script.

There is no means of anticipating the browser denying access due to this security restriction.

CX UI Capture j2 continues to function normally. However, actions that are executed in the iFrame cannot be monitored.

Some browsers record an error message in the console. For purposes of CX UI Capture j2 functions that are described above, this message can be ignored and does not adversely impact either CX UI Capture j2 or your application.

Upgrading the library

How do I determine which version of the library I am using?

For IBM Tealeaf CX UI Capture j2, library version information is provided in the `X-Tealeaf` header.

See "UI Capture j2 Overview".

How do I determine which version of the `TealeafTarget.jsp` file I am using?

Through the browser, navigate to the `TealeafTarget.jsp` file on your website by using the `version` query parameter.

The following URL is an example.

```
http://mydomain.com/mySite/TealeafTarget.jsp?version
```

Note: If this URL does not return the version information, you do not have the latest version.

When do I upgrade?

Whenever possible, use the latest version of IBM Tealeaf CX UI Capture j2.

If you are experiencing UI capture issues, you must upgrade to the latest version of IBM Tealeaf CX UI Capture j2 since it contains the most recent improvements.

You can integrate the CX UI Capture j2 upgrade into the existing maintenance and deployment processes for your application.

Note: Before you upgrade, review the *IBM Tealeaf CX UI Capture j2 Release Notes* to assess its suitability for your web application.

How do I find out about available updates?

1. Acquire the version number for your current installation of IBM Tealeaf CX UI Capture j2. See “How do I determine which version of the library I am using?.”
2. Compare the above number to the latest available version. For more information on downloading IBM Tealeaf, see IBM Passport Advantage Online.

Version information is also published in the general Release Notes in the IBM Tealeaf Online Help.

Chapter 8. IBM Tealeaf documentation and help

IBM Tealeaf provides documentation and help for users, developers, and administrators.

Viewing product documentation

All IBM Tealeaf product documentation is available at the following website:

https://tealeaf.support.ibmcloud.com/

Use the information in the following table to view the product documentation for IBM Tealeaf:

Table 3. Getting help

To view...	**Do this...**
Product documentation	On the IBM Tealeaf portal, go to **?** > **Product Documentation**.
Help for a page on the IBM Tealeaf Portal	On the IBM Tealeaf portal, go to **?** > **Help for This Page**.

Available documents for IBM Tealeaf products

Use the following table to view a list of available documents for all IBM Tealeaf products:

Table 4. Available documentation for IBM Tealeaf products

IBM Tealeaf products	**Available documents**
IBM Tealeaf CX	• *IBM Tealeaf Customer Experience Overview Guide* • *IBM Tealeaf CX Client Framework Data Integration Guide* • *IBM Tealeaf CX Configuration Manual* • *IBM Tealeaf CX Cookie Injector Manual* • *IBM Tealeaf CX Databases Guide* • *IBM Tealeaf CX Event Manager Manual* • *IBM Tealeaf CX Glossary* • *IBM Tealeaf CX Installation Manual* • *IBM Tealeaf CX PCA Manual* • *IBM Tealeaf CX PCA Release Notes*

© Copyright IBM Corp. 1999, 2014

Table 4. Available documentation for IBM Tealeaf products (continued)

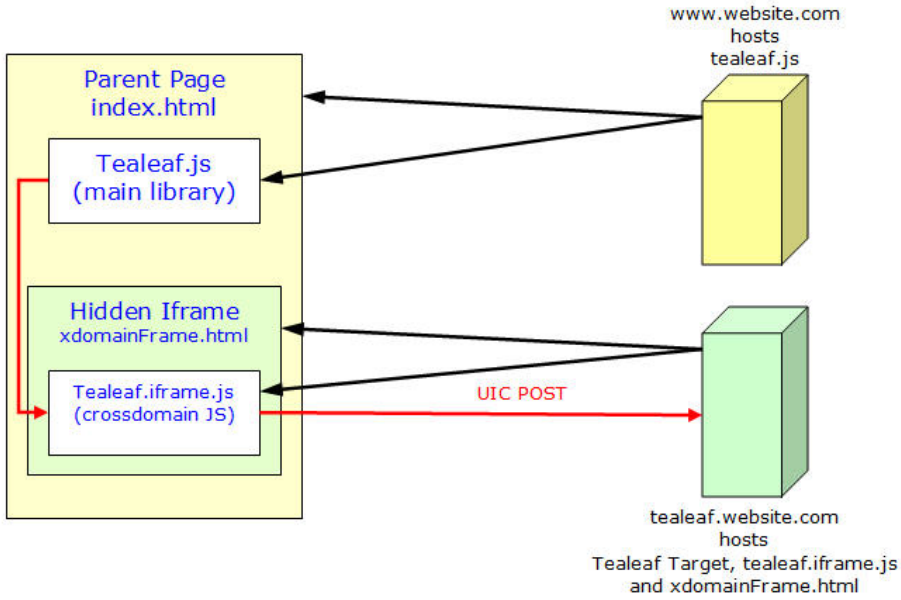
IBM Tealeaf products	Available documents
IBM Tealeaf CX	• *IBM Tealeaf CX RealiTea Viewer Client Side Capture Manual* • *IBM Tealeaf CX RealiTea Viewer User Manual* • *IBM Tealeaf CX Release Notes* • *IBM Tealeaf CX Release Upgrade Manual* • *IBM Tealeaf CX Support Troubleshooting FAQ* • *IBM Tealeaf CX Troubleshooting Guide* • *IBM Tealeaf CX UI Capture j2 Guide* • *IBM Tealeaf CX UI Capture j2 Release Notes*
IBM Tealeaf cxImpact	• *IBM Tealeaf cxImpact Administration Manual* • *IBM Tealeaf cxImpact User Manual* • *IBM Tealeaf cxImpact Reporting Guide*
IBM Tealeaf cxConnect	• *IBM Tealeaf cxConnect for Data Analysis Administration Manual* • *IBM Tealeaf cxConnect for Voice of Customer Administration Manual* • *IBM Tealeaf cxConnect for Web Analytics Administration Manual*
IBM Tealeaf cxOverstat	*IBM Tealeaf cxOverstat User Manual*
IBM Tealeaf cxReveal	• *IBM Tealeaf cxReveal Administration Manual* • *IBM Tealeaf cxReveal API Guide* • *IBM Tealeaf cxReveal User Manual*
IBM Tealeaf cxVerify	*IBM Tealeaf cxVerify Administration Manual*
IBM Tealeaf cxView	*IBM Tealeaf cxView User Manual*
IBM Tealeaf CX Mobile	• *IBM Tealeaf CX Mobile Android Logging Framework Guide* • *IBM Tealeaf Android Logging Framework Release Notes* • *IBM Tealeaf CX Mobile Administration Manual* • *IBM Tealeaf CX Mobile User Manual* • *IBM Tealeaf CX Mobile iOS Logging Framework Guide* • *IBM Tealeaf iOS Logging Framework Release Notes*

Appendix A. Cross-domain communication

In normal operation, CX UI Capture j2 POSTs the captured data to the configured URL using the protocol (for example, http or https), domain (www.website.com), and port of the parent page. This is the preferred method of deployment.

In some cases, it may be required to send the CX UI Capture j2 POST request to a different server than the parent page. For example, the site may have deployed with a dedicated server to host the target page that has its own subdomain (tealeaf.website.com).

This can be visualized as follows:



To perform this type of POST, the website must be configured as follows.

1. Deploy the following to the cross-domain server.
   - IBM Tealeaf target page.
   - Tealeaf cross-domain JavaScript from the CX UI Capture j2 package.
   - A minimal html source to include the JS and set the document domain to enable cross-domain communication. An example of the html (xdomainFrame.html) is provided below.

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        // Ensure the document domain is set to the sub-domain.
        document.domain = "website.com";
    </script>
    <script src="tealeaf.iframe.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

© Copyright IBM Corp. 1999, 2014

2. Add an iframe element to the pages and set the document domain to the subdomain to enable cross-domain communication. Mark the iframe element with a CSS style of display: none so it does not impact the rendered output of the page.

```
<iframe id="uicXdomain"
src="tealeaf.website.com/xdomainFrame.html" width="0"
height="0" tabindex="-1" title="empty" style="display:
none"></iframe>

<script type="text/javascript">
    // Ensure the document domain is set to the sub-domain.
    document.domain = "website.com";
</script>
```

3. Configure the main UI Capture library to set the `crossDomainEnabled` to true and specify the `crossDomainFrameSelector` to correspond to the hidden iframe added in step 2.

Appendix B. Variations between jQuery and W3c flavors of CX UI Capture j2

There are several variations between the jQuery and W3C flavors of CX UI Capture j2.

Change event

In legacy Internet Explorer, the change event does not bubble. However, jQuery normalizes this behavior and causes the change to bubble on all versions of Internet Explorer. Therefore, when using the jQuery flavor, the change event listener does not need to be configured to attach to individual elements.

For the same reason, on pages with frames or iframes, the change event cannot be tracked on elements contained within these frame elements (in legacy Internet Explorer) when using the W3C flavor of CX UI Capture j2. A workaround for this type of situation is to include the W3C flavor of the CX UI Capture j2 library in the frame page itself.

Event delegation support in the jQuery flavor of CX UI Capture j2

The jQuery flavor of the CX UI Capture j2 supports delegate targets as implemented by the jQuery library. This is not supported by the W3C flavor.

Specifying a delegate target works for the events that bubble from the innermost element in the document where they occur all the way up to the body and the document element. In Internet Explorer 8 and lower, a few events such as change and submit do not natively bubble but jQuery patches these to bubble and create consistent cross-browser behavior.

To use event delegation, the `delegateTarget` property should be added to the event entry as follows.

```
{ name: "change", delegateTarget: "document", target: "input.selected" }
```

The `delegateTarget` points to the document or a selector that resolves to a single target element. The event is not registered when it occurs directly on the delegate target element, but only for descendants (inner elements) that match the target selector.

For example, consider the following markup.

```
<div id="container">
  <input id="x" type="text" />
    <input id="y" type="text" />
    <input id="z" type="text" />
</div>
```

Without event delegation, you would have to specify the focus and blur event handlers as follows.

```
{ name: "focus", target: "input" }
{ name: "blur",  target: "input" }
```

This results in event handlers being attached to each input element. With event delegation, one can specify the same as the following.

© Copyright IBM Corp. 1999, 2014

```
{ name: "focusin",  delegateTarget: "#container", target: "input" }
{ name: "focusout", delegateTarget: "#container", target: "input" }
```

This would result in attaching a single event handler to the `delegateTarget` element, which is the container `div`.

Note: Since focus & blur do not bubble, they need to be replaced with `focusin` and `focusout`, which are synthetic events that are provided by jQuery that do bubble.

As another example, consider the following markup.

```
<div id="menu">
    <a id="item1" href="#item1">Item 1</a>
    <a id="item2" href="#item2">Item 2</a>
    <a id="item3" href="#item3">Item 3</a>
    <a id="item4" href="#item4">Item 4</a>
    <a id="item5" href="#item5">Item 5</a>
</div>
```

To monitor `mouseover` events on the menu items one could specify the following configuration.

```
{ name: "mouseover", target: "div#menu > a" }
```

This would attach an event handler to each menu item. With event delegation, one can achieve a similar result by specifying the following.

```
{ name: "mouseover", delegateTarget: "#menu", target: "a" }
```

This would attach a single event handler to the `delegateTarget` element, which is the `div`.

Delegated events have the advantage that they can process events from descendant elements that are added to the document later. By picking an element that is guaranteed to be present at the time the delegated event handler is attached. You can use delegated events to avoid the need to frequently attach and remove event handlers. This element could be the container element of a view in a Model-View-Controller design, for example, or document if the event handler wants to monitor all bubbling events in the document.

In addition to their ability to handle events on descendant elements that are not yet created, another advantage of delegated events is their potential for much lower overhead when many elements must be monitored. On a data table with thousands of elements, a delegated-events approach attaches an event handler to only one element, the container, and the event only needs to bubble up one or two levels at the most. This results in improved performance.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 1999, 2014

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Bay Area Lab
1001 E Hillsdale Boulevard
Foster City, California 94404
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Privacy Policy Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer's website terms of use (e.g. privacy policy) which includes a link to IBM's and Client's data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor's computer by IBM on the Client's behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client's behalf on website visitor's devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Online Privacy Statement at: http://www.ibm.com/privacy/details/us/en section entitled "Cookies, Web Beacons and Other Technologies."

Printed in USA